BỘ GIAO THÔNG VẬN TẢI
TRƯỜNG ĐẠI HỌC HÀNG HẢI
BỘ MÔN: KỸ THUẬT MÁY TÍNH
KHOA: CÔNG NGHỆ THÔNG TIN

BÀI GIẢNG

HỆ THỐNG NHÚNG

TÊN HỌC PHẦN : **HỆ THỐNG NHÚNG**
MÃ HỌC PHẦN : **17312**
TRÌNH ĐỘ ĐÀO TẠO : **ĐẠI HỌC CHÍNH QUY**
DÙNG CHO SV NGÀNH : **CÔNG NGHỆ THÔNG TIN**

HẢI PHÒNG - 2010

MỤC LỤC

YÊU CẦU VÀ NỘI DUNG CHI TIẾT

Tên học phần: **Hệ thống nhúng** *Loại học phần:* **3**
Bộ môn phụ trách giảng dạy: **Kỹ thuật máy tính** *Khoa phụ trách:* **CNTT**
Mã học phần: **17312** *Tổng số TC:* **3**

TS tiết	Lý thuyết	Thực hành/Xemina	Tự học	Bài tập lớn	Đồ án môn học
45	45	0	0	x	0

Điều kiện tiên quyết:

Sinh viên phải học xong các học phần sau mới được đăng ký học phần này:

Kiến trúc máy tính, Điện tử số, Mạch và tín hiệu, Kỹ thuật Vi xử lí, Nguyên lý hệ điều hành,..

Mục tiêu của học phần:

Cung cấp các kiến thức cơ bản về về các kiến trúc và mô hình của hệ thống nhúng

Áp dụng xây dựng các hệ thống nhúng cơ bản dựa trên các thiết bị, các họ vi xử lý thông dụng

Nội dung chủ yếu

Chương 1: Tổng quan

Chương 2: Hệ thống phần cứng

Chương 3: Phần mềm nhúng

Chương 4: Thiết kế hệ thống nhúng theo các họ Vi xử lí

Nội dung chi tiết:

TÊN CHƯƠNG MỤC	PHÂN PHỐI SỐ TIẾT				
	TS	**LT**	**BT**	**TH**	**KT**
Chương 1: Tổng quan	**8**	**8**			
1.1. Tổng quan		1			
1.2. Kiến trúc hệ thống nhúng		2			
1.3. Thiết kế hệ thống nhúng		2			
1.4. Các mô hình hệ thống nhúng		2			
1.5. Các chuẩn		1			
Chương 2: Hệ thống phần cứng	**10**	**9**			**1**
2.1. Tổng quan		1			
2.2. Hệ Vi xử lí		3			
2.2.1. Tổng quan					
2.2.2. Kiến trúc vi xử lí trong các hệ thống nhúng					
2.2.3. Sự thực thi					
2.3. Bộ nhớ		2			
2.3.1. Tổng quan					
2.3.2. Bộ nhớ ROM					
2.3.3. Bộ nhớ RAM					
2.3.4. Quản lí bộ nhớ					

TÊN CHƯƠNG MỤC	PHÂN PHỐI SỐ TIẾT				
	TS	**LT**	**BT**	**TH**	**KT**
2.4. Thiết bị ngoại vi		2			
2.4.1. Tổng quan					
2.4.2. Vào ra nối tiếp					
2.4.3. Vào ra song song					
2.5. BUS		1			1
Chương 3: Phần mềm nhúng	**9**	**8**			**1**
3.1. Tổng quan		1			
3.2. Trình điều khiển thiết bị		2			
3.2.1. Tổng quan					
3.2.2. Ngắt					
3.2.3. Bộ nhớ					
3.2.4. BUS					
3.3. Hệ điều hành trong các hệ thống nhúng		4			
3.3.1. Tổng quan					
3.3.2. Tiến trình					
3.3.3. Quản lí tiến trình					
3.3.4. Quản lí bộ nhớ					
3.3.5. Quản lí thiết bị ngoại vi					
3.4. Phần mềm ứng dụng		2			1
Chương 4: Thiết kế hệ thống nhúng theo các họ VXL	**16**	**16**			**BTL**
4.1. Tổng quan		1			
4.2. Họ vi xử lí AT89C		5			
4.2.1. Tổng quan					
4.2.2. Kiến trúc họ					
4.2.3. Tập lệnh					
4.2.4. Sự thực thi					
4.2.5. Thiết kế ứng dụng					
4.3. Họ vi xử lí AVR		5			
4.3.1. Tổng quan					
4.3.2. Kiến trúc họ vi xử lý AVR					
4.3.3. Tập lệnh					
4.3.4. Sự thực thi					
4.3.5. Thiết kế ứng dụng					*1*
4.4. Họ vi xử lí ARM		5			
4.4.1. Tổng quan					
4.4.2. Kiến trúc họ					
4.4.3. Tập lệnh					
4.4.4. Sự thực thi					
4.4.5. Thiết kế ứng dụng					

Nhiệm vụ của sinh viên :

Tham dự các buổi thuyết trình của giáo viên, tự học, tự làm bài tập do giáo viên giao, tham dự các bài kiểm tra định kỳ và cuối kỳ, hoàn thành bài tập lớn theo yêu cầu.

Tài liệu học tập :

- Al.M.Zied , ***Embedded System Architecture***, NXB Elsevier
- John Catsoulis, ***Designing Embedded Hardware***, NXB O'Reilly
- Ken Arnold, ***Embedded Controller Hardware Design***, NXB LLH Technology
- Dhananjay V.Gadre, ***Programming And Customizing The AVR Microcontroller***, NXB Mc.Graw Hill
- Steve Furber, ***ARM System On Chip Architecture***, NXB Dorling Kindersley

Hình thức và tiêu chuẩn đánh giá sinh viên:

- Đánh giá dựa trên tình hình tham dự buổi học trên lớp, các buổi thực hành, điểm kiểm tra thường xuyên và điểm kết thúc học phần.
- Hình thức thi cuối kỳ : thi viết + kiểm tra vấn đáp BTL

Thang điểm: Thang điểm chữ A, B, C, D, F

Điểm đánh giá học phần Z = 0.3X + 0.7Y.

Bài giảng này là tài liệu **chính thức và thống nhất** của Bộ môn Kỹ thuật máy tính, Khoa Công nghệ Thông tin và được dùng để giảng dạy cho sinh viên.

Ngày phê duyệt: 15 / 6 / 2010

Trưởng Bộ môn: ThS. Ngô Quốc Vinh

CHƯƠNG 1 - TỔNG QUAN

1.1 Tổng quan

Hệ điều khiển nhúng là một môn học mới nhằm cung cấp kiến thức cho sinh viên về khả năng phân tích và thiết kế hệ thống điều khiển và thông minh hoá hệ thống theo chức năng theo giải pháp công nghệ. Thiết kế thực thi điều khiển trên nền phần cứng nhúng.

Kỷ nguyên công nghệ mới đã và đang tiếp tục phát triển không ngừng nhằm thông minh hoá hiện đại hoá thông suốt các hệ thống. Có thể nói đánh dấu sự ra đời và phát triển của hệ nhúng trước tiên phải kể đến sự ra đời của các bộ vi xử lý, vi điều khiển. Nó được đánh dấu bởi sự ra đời của Chip vi xử lý đầu tiên 4004 vào năm 1971 cho mục đích tính toán thương mại bởi một công ty Nhật bản Busicom và sau đó đã được chắp cánh và phát triển vượt bậc bởi Intel để trở thành các bộ siêu xử lý như các Chip được ứng dụng cho PC như ngày nay. Thập kỷ 80 có thể được coi là khởi điểm bắt đầu kỷ nguyên của sự bùng nổ về phát triển các hệ nhúng. Từ đó khởi nguồn cho làn sóng ra đời của hàng loạt các chủng loại vi xử lý và gắn liền là các hệ nhúng để thâm nhập rộng khắp trong các ứng dụng hàng ngày của cuộc sống chúng ta ví dụ như, các thiết bị điện tử sử dụng cho sinh hoạt hàng ngày (lò vi sóng, TV, tủ lạnh, máy giặt, điều hoà ...) và văn phòng làm việc (máy fax, máy in, máy điện thoại...)... Các bộ vi xử lý và phần mềm cũng ngày càng được sử dụng rộng rãi trong rất nhiều các hệ thống nhỏ. Các loại vi xử lý được sử dụng trong các hệ thống nhúng hiện nay đã vượt xa so với PC về số lượng chủng loại (chiếm đến 79% số các vi xử lý đang tồn tại [2]) và vẫn còn tiếp tục phát triển để nhằm đáp ứng và thoả mãn rất nhiều ứng dụng đa dạng. Trong số đó vẫn còn ứng dụng cả các Chip vi xử lý 8 bit, 16 bit và hiện nay chủ yếu vẫn là 32 bit (chiếm khoảng 75%). Gắn liền với sự phát triển phần cứng, phần mềm cũng đã phát triển với tốc độ nhanh không thua kém thậm chí sẽ tăng nhanh hơn rất nhiều theo sự phát triển hệ nhúng. Ta xem xét một số khái niệm sau.

1.1.1 Hệ thống nhúng

Vậy thế nào là một hệ nhúng? Trong thế giới thực của chúng ta bất kỳ một thiết bị hay hệ thống điện/điện tử có khả năng xử lý thông tin và điều khiển đều có thể tiềm ẩn trong đó một thiết bị hay hệ nhúng, ví dụ như các thiết bị truyền thông, thiết bị đo lường điều khiển, các thiết bị phục vụ sinh hoạt hàng ngày như lò vi sóng, máy giặt, camera…Rất dễ dàng để có thể kể ra hàng loạt các thiết bị hay hệ thống như vậy đang tồn tại quanh ta, chúng là hệ nhúng. Vậy hệ nhúng thực chất là gì và nên hiểu thế nào về hệ nhúng? Hiện nay cũng chưa có một định nghĩa nào thực sự thoả đáng để được chuẩn hoá và thừa nhận rộng rãi cho hệ nhúng mà vẫn chỉ là những khái niệm diễn tả về chúng thông qua những đặc thù chung. Tuy nhiên ở đây chúng ta có thể hiểu hệ nhúng là một phần hệ thống xử lý thông tin nhúng trong các hệ thống lớn, phức hợp và độc lập ví dụ như trong ôtô, các thiết bị đo lường, điều khiển, truyền thông và thiết bị thông minh nói chung. Chúng là những tổ hợp của phần cứng và phần mềm để thực hiện một hoặc một nhóm chức năng chuyên biệt, cụ thể (Trái ngược với máy tính PC mà chúng ta thường thấy được sử dụng không phải cho một chức năng mà là rất nhiều chức năng hay phục vụ chung cho nhiều mục đích). PC thực chất lại là một hệ thống lớn, tổ hợp của nhiều hệ thống nhúng ví dụ như card màn hình, âm thanh, modem, ổ cứng, bàn phím…Chính điều này làm chúng ta dễ lúng túng nếu được hỏi nên hiểu thế nào về PC, có phải là hệ nhúng hay không.

1.1.2 Hệ thống thời gian thực

Trong các bài toán điều khiển và ứng dụng chúng ta rất hay gặp thuật ngữ "thời gian thực". Thời gian thực có phải là thời gian phản ánh về độ trung thực của thời gian hay không? Thời gian thực có phải là hiển thị chính xác và đồng bộ theo đúng như nhịp đồng hồ đếm thời gian hay không? Không phải hoàn toàn như vậy! Thực chất, theo cách hiểu nếu nói trong các hệ thống kỹ thuật đặc biệt các hệ thống yêu cầu khắt khe về sự ràng buộc thời gian, thời gian

thực được hiểu là yêu cầu của hệ thống phải đảm bảo thoả mãn về tính tiền định trong hoạt động của hệ thống. Tính tiền định nói lên hành vi của hệ thống thực hiện đúng trong một khung thời gian cho trước hoàn toàn xác định. Khung thời gian này được quyết định bởi đặc điểm hoặc yêu cầu của hệ thống, có thể là vài giây và cũng có thể là vài nano giây hoặc nhỏ hơn nữa. Ở đây chúng ta phân biệt yếu tố thời gian gắn liền với khái niệm về thời gian thực. Không phải hệ thống thực hiện rất nhanh là sẽ đảm bảo được tính thời gian thực vì nhanh hay chậm hoàn toàn là phép so sánh có tính tương đối vì mili giây có thể là nhanh với hệ thống điều khiển nhiệt nhưng lại là chậm đối với các đối tượng điều khiển điện như dòng, áp…. Hơn thế nữa nếu chỉ nhanh không thì chưa đủ mà phải đảm bảo duy trì ổn định bằng một cơ chế hoạt động tin cậy. Chính vì vậy hệ thống không kiểm soát được hoạt động của nó (bất định) thì không thể là một hệ thống đảm bảo tính thời gian thực mặc dù hệ thống đó có thể cho đáp ứng rất nhanh, thậm chí nhanh hơn rất nhiều so với yêu cầu đặt ra. Một ví dụ minh hoạ tiêu biểu đó là cơ chế truyền thông dữ liệu qua đường truyền chuẩn Ethernet truyền thống, mặc dù ai cũng biết tốc độ truyền là rất nhanh nhưng vẫn không phải hệ hoạt động thời gian thực vì không thoả mãn tính tiền định trong cơ chế truyền dữ liệu (có thể là rất nhanh và cũng có thể là rất chậm nếu có sự canh tranh và giao thông đường truyền bị nghẽn).

Người ta phân ra làm hai loại đối với khái niệm thời gian thực là cứng (hard real time) và mềm (soft real time). Thời gian thực cứng là khi hệ thống hoạt động với yêu cầu thoả mãn sự ràng buộc trong khung thời gian cứng tức là nếu vi phạm thì sẽ dẫn đến hoạt động của toàn hệ thống bị sai hoặc bị phá huỷ. Ví dụ về hoạt động điều khiển cho một lò phản ứng hạt nhân, nếu chậm ra quyết định có thể dẫn đến thảm hoạ gây ra do phản ứng phân hạch và dẫn đến bùng nổ cả hệ thống. Thời gian thực mềm là khi hệ thống hoạt động với yêu cầu thoả mãn ràng buộc trong khung thời gian mềm, nếu vi phạm và sai lệch nằm trong khoảng cho phép thì hệ thống vẫn có thể hoạt động được và chấp nhận được. Ví dụ như hệ thống phát thanh truyền hình, nếu thông tin truyền đi từ trạm phát tới người nghe/nhìn chậm một vài giây thì cũng không ảnh hưởng đáng kể đến tính thời sự của tin được truyền đi và hoàn toàn được chấp nhận bởi người theo dõi. Thực tế thấy rằng hầu hết hệ nhúng là các hệ thời gian thực và hầu hết các hệ thời gian thực là hệ nhúng. Điều này phản ánh mối quan hệ mật thiết giữa hệ nhúng và thời gian thực và tính thời gian thực đã trở thành như một thuộc tính tiêu biểu của hệ nhúng. Vì vậy hiện nay khi đề cập tới các hệ nhúng người ta đều nói tới đặc tính cơ bản của nó là tính thời gian thực.

1.1.3 Lĩnh vực ứng dụng của hệ thống nhúng

Chúng ta có thể kể ra được rất nhiều các ứng dụng của hệ thống nhúng đang được sử dụng hiện nay, và xu thế sẽ còn tiếp tục tăng nhanh. Một số các lĩnh vực và sản phẩm thị trường rộng lớn của các hệ nhúng có thể được nhóm như sau:

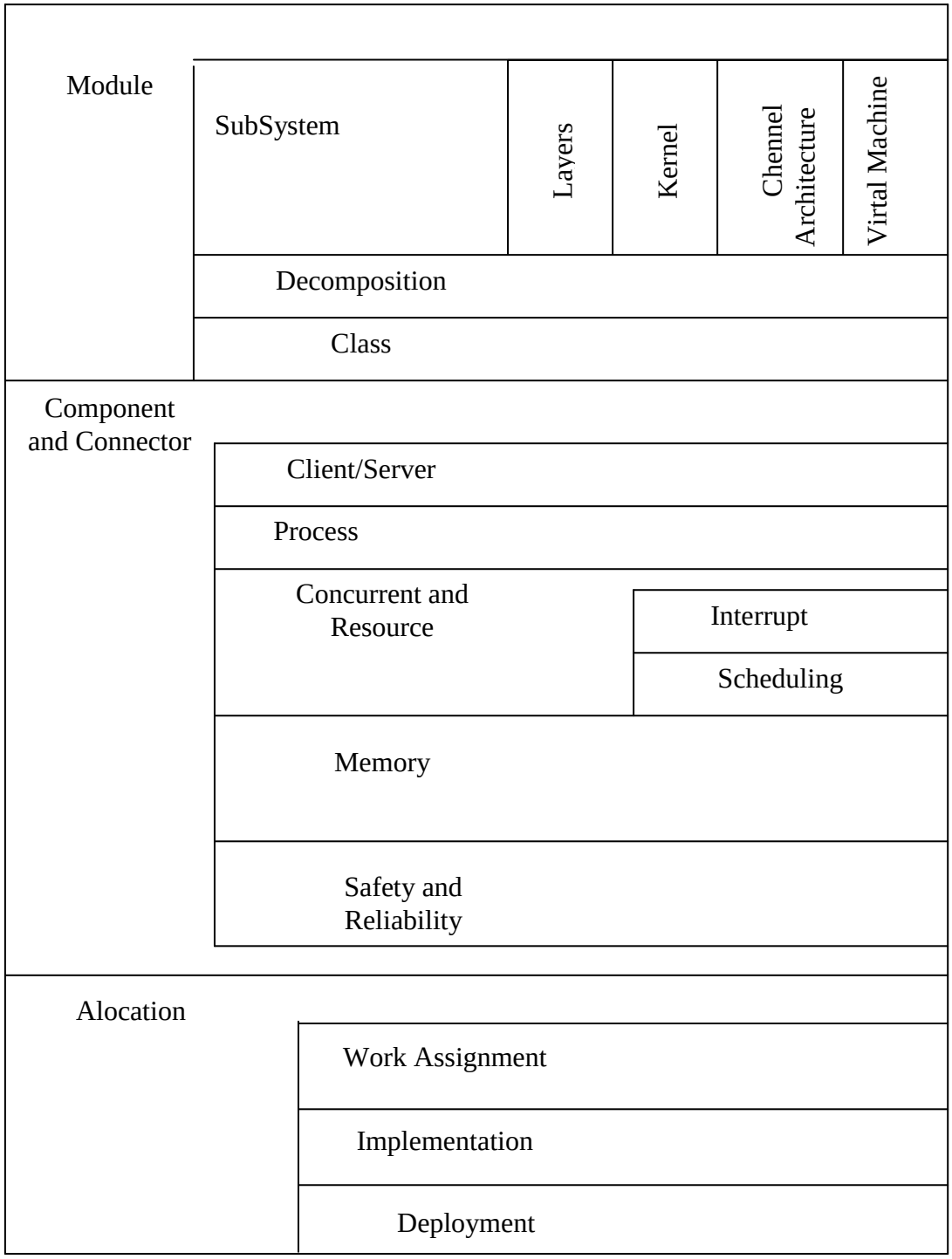
- Các thiết bị điều khiển
- Ôtô, tàu điện
- Truyền thông
- Thiết bị y tế
- Hệ thống đo lường thẩm định
- Toà nhà thông minh
- Thiết bị trong các dây truyền sản xuất
- Rôbốt
- ...

1.2 Kiến trúc hệ thống nhúng

Kiến trúc của một hệ thống nhúng thể hiện ở mức độ trong suốt của các thiết bị nhúng, đó là các hệ thống nhúng thông thường sẽ không thể hiện các thông tin cài đặt cụ thể như mã nguồn hoặc các chi tiết về mạch điện. Tại mỗi mức của kiến trúc, các thành phần phần cứng

và phần mềm sẽ thể hiện một số những elements, là những đơn vị tương tác với những thành phần khác. Elements là các thể hiện của phần cứng và phần mềm mà những thông tin cài đặt cụ thể được ẩn đi, elements chỉ cho ta biết những thông tin về hành vi của chúng. Ta có internal và external elemént, internal element liên kết với các thiết bị nhúng còn external elemements liên kết với các internal elements. Tóm lại, kiến trúc của một hệ thống nhúng bao gồm các thành phần của hệ thống đó. Các elements liên kết với hệ thống, thuộc tính của từng thành phần elements cụ thể và mối quan hệ giữa các elements.

Thông tin về các mức của kiến trúc hệ thống nhúng được biểu diễn dưới dạng các cấu trúc. Một cấu trúc có thể biểu diễn một phần kiến trúc, đồng thời bao gồm tập hợp các thành phần, thuộc tính và các mối liên hệ giữa các thành phần. Mỗi cấu trúc do đó là một snapshot của hệ thống phần cứng/phần mềm tại thời điểm thiết kế hoặc thực thi. Cho trước một môi trường thực thi và một tập hợp các thành phần. Do rất khó để có thể dùng một snapshot mô tả tất cả những độ phức tạp của cả hệ thống, thông tin về kiến trúc thường được tạo ra từ nhiều cấu trúc. Tất cả mọi cấu trúc trong một kiến trúc đều được kế thừa và liên quan đến các cấu trúc khác. Bảng sau mô tả một số các cấu trúc cơ bản nhất và đưa ra giải thích về sự liên quan giữa những thành phần này.



Hình 1.1 Các kiến trúc cơ bản của hệ thống nhúng

Trong đó:

- Module: là những thành phần được định nghĩa với những chức năng khác nhau, là những đơn vị phần mềm/phần cứng cần thiết để hệ thống có thể hoạt động đúng. Cấu trúc mô tả với những module này thường được sử dụng để giới thiệu một sản phẩm nào đó.
- SubSystem: biểu diễn hình ảnh của một module tại thời điểm thực thi trong đó có sự liên kết hoạt động giữa các module với nhau

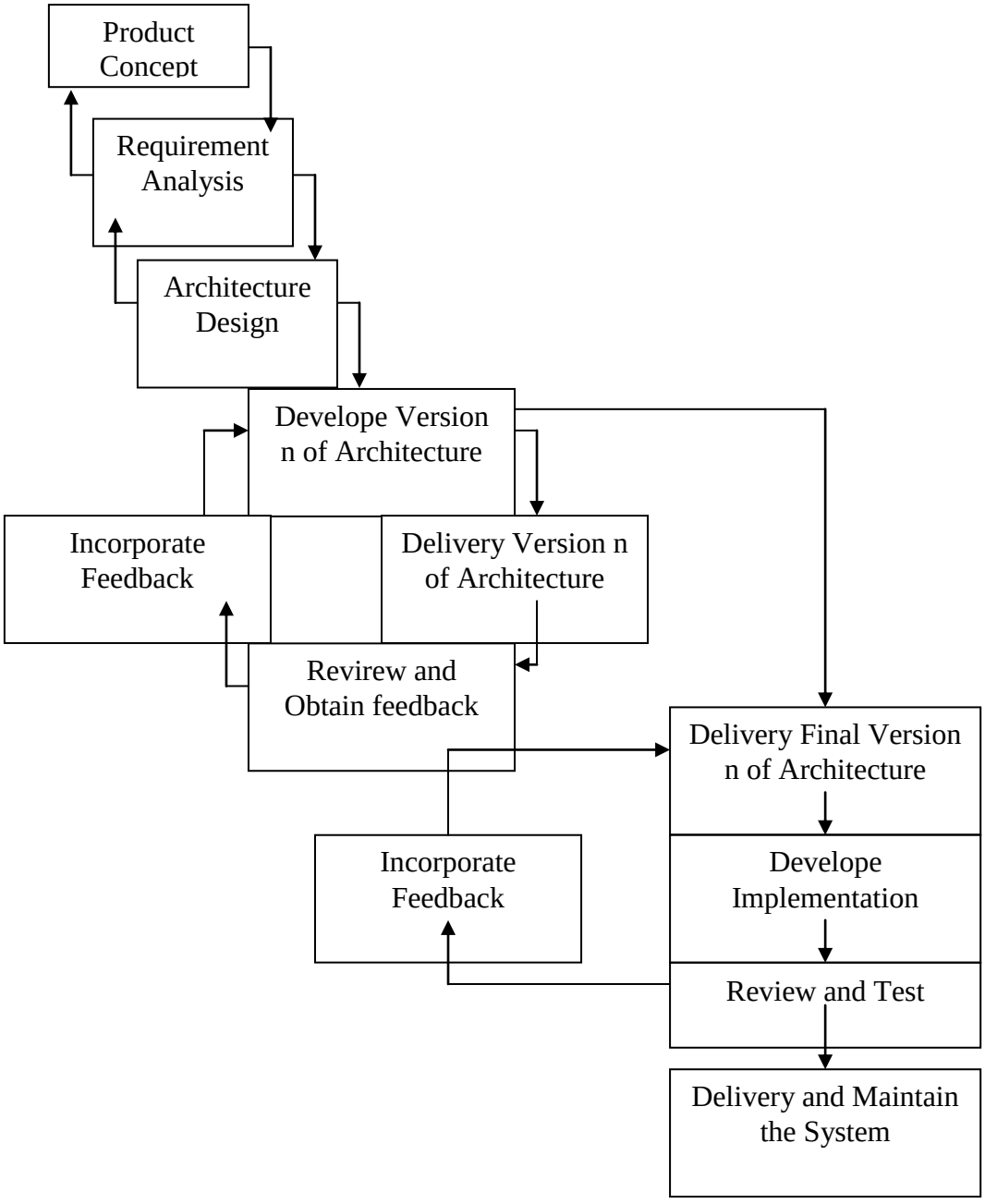
- Layers: một kiểu của SubSystem trong đó các module được biểu diễn dưới dạng các lớp, module ở lớp trên sẽ sử dụng các dịch vụ được cung cấp bởi module ở lớp dưới.
- Kernel: cấu trúc biểu diễn các module có sử dụng các dịch vụ của kernel hoặc được quản lý bởi kernel.
- Channel Architecture: cấu trúc biểu diễn các module dạng chuỗi, mô tả những sự thay đổi trạng thái của module trong quá trình hoạt động.
- Virtual Machine: cấu trúc biểu diễn các module sử dụng các dịch vụ của một máy ảo.
- Decomposition: một kiểu cấu trúc module trong đó một số module là các module con của module khác, thể hiện trong các mối quan hệ giữa những module này. Cấu trúc này thường được sử dụng để xác định các tài nguyên, quản lý dự án, quản lý dữ liệu ...
- Class: là một kiểu cấu trúc biểu diễn các đơn vị phần mềm trong đó các module được tham chiếu là các lớp, và quan hệ giữa chúng được định nghĩa dựa theo mô hình hướng đối tượng trong đó lớp này kế thừa từ lớp khác hoặc là một thể hiện của lớp cha.
- Component or Connector: các cấu trúc này bao gồm các thành phần hoặc là các components ví dụ như các đơn vị xử lý phần cứng, phần mềm, bộ xử lý, máy ảo... hoặc các Connector (các đơn vị kết nối giữa các thành phần, như hệ thống bus phần cứng hoặc hệ thống thông điệp phần mềm)
- Client/Server: kiểu cấu trúc mô tả hệ thống tại thời điểm thực thi với các thành phần là clients hoặc server và các connector là các cơ chế kết nối (như giao thức, thông điệp, gói tin ...) được sử dụng để liên kết giữa clients và server.
- Process: cấu trúc này mô tả phần mềm của hệ thống trong đó chứa hệ điều hành, các thành phần khác như các process và các tiến trình và các liên kết của chúng.
- Concurrency and Resource: cấu trúc này mô tả một snapshot của hệ thống bao gồm OS, và các thành phần trong đó. Cấu trúc này được sử dụng trong việc quản lý tài nguyên và để xác định xem có vấn đề gì với việc chia sẻ các tài nguyên cũng như các tiến trình có thể thực thi song song hay không.
- Interrupt: cấu trúc mô tả các cơ chế xử lý ngắt trong hệ thống.
- Scheduling: cấu trúc mô tả cơ chế lập lịch và quản lý tiến trình trong hệ thống.
- Memory: mô tả hình ảnh của bộ nhớ và các thành phần dữ liệu trong bộ nhớ cũng như mô tả các cơ chế quản lý bộ nhớ của hệ thống.
- Safety and Reliability: cấu trúc mô tả hệ thống tại thời điểm thực thi trong đó biểu diễn những thành phần dư thừa và những mối liên hệ của chúng để đánh giá độ an toàn và tin cậy của cả hệ thống.
- Allocation: cấu trúc mô tả mối liên hệ giữa các thành phần phần cứng/phần mềm và các thực thể của môi trường bên ngoài.
- Work Assignment: cấu trúc này gán cho các nhóm phát triển những công việc (các module) cần thực hiện. Nó được sử dụng trong việc quản lý dự án.
- Implementation: đây là cấu trúc phần mềm chỉ ra vị trí mà phần mềm đó trong hệ thống file.
- Deployment: cấu trúc này mô tả hệ thống tại thời điểm thực thi với các thành phần của cả phần cứng và phần mềm và mối liên hệ giữa chúng.

1.3 Thiết kế hệ thống nhúng

Ta có thể sử dụng một số mô hình sau để mô tả chu kỳ thiết kế các hệ thống nhúng. Một số mô hình là cơ sở, các mô hình khác được hình thành dựa trên các mô hình cơ sở này.

- Mô hình big-bang: trong mô hình thiết kế này, ta không có khái niệm về kế hoạch hay quá trình trong cả quá trình phát triển hệ thống.
- Mô hình code-and-fix: đầu tiên các yêu cầu về sản phẩm được làm rõ, sau đó việc thực hiện mã lệnh được tiến hành dựa trên các mô tả yêu cầu này, tiếp theo mã lệnh được thực thi và nếu có lỗi thì lại trở về bước trước đó, nghĩa là thực hiện lại mã lệnh.
- Mô hình waterfall: Quá trình phát triển sản phẩm được chia thành từng bước, kết quả của bước trước sẽ là dữ liệu của bước sau.
- Mô hình Spiral: cũng dựa trên việc phân chia thành từng bước như waterfall, tuy nhiên trong mỗi một quá trình, các phản hồi của người dùng hoặc người phát triển khác được tiếp thu và được tích hợp lại vào trong quá trình phát triển tiếp theo.

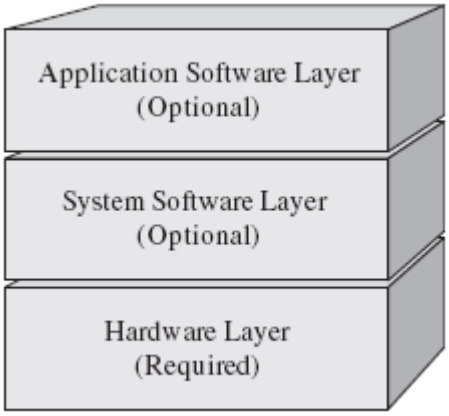
Ta xét mô hình Embedded Design and Development Lifecycle Model sau:



Hình 1.2 Mô hình Embedded Design and Development Lifecycle

Mô hình này dựa trên sự kết hợp giữa waterfall và Spiral, ta sẽ xét chi tiết mô hình trong những phần sau

1.4 Mô hình hệ thống nhúng



Hình 1.3 Mô hình hệ thống nhúng

Hình trên chỉ ra rằng tất cả các hệ thống nhúng đều có chung một thành phần giống nhau ở tầng cao nhất, đó là chúng đều có ít nhất một lớp (phần cứng) hoặc nhiều lớp (phần cứng, phần mềm và ứng dụng) trong đó chứa tất cả các components. Phần cứng bao gồm tất cả những thành phần vật lý có trên mạch nhúng, phần mềm và các ứng dụng bao gồm tất cả những thành phần logic có trong hệ thống nhúng.

Mô hình tham chiếu trên là cách biểu diễn phân lớp của kiến trúc các hệ thống nhúng từ đó các cấu trúc module có thể được suy ra. Nếu bỏ qua những sự khác nhau giữa các thiết bị trong bảng trên, có thể nói rằng kiến trúc của mọi hệ thống được biểu diễn thông qua việc thể hiện và nhóm các thành phần được gọi là các lớp. Ta cũng chú ý là lớp không chỉ là khái niệm đặc thù của riêng hệ thống nhúng mà còn là của nhiều hệ thống khác. Đây là công cụ hữu ích để mô hình hóa sự kết hợp giữa hàng trăm, có thể hàng ngàn thành phần trong thiết kế hệ thống nhúng. Những nguyên nhân chính khiến chúng trở lên hữu ích là:

Thể hiện được các thành phần quan trọng và các hành vi của chúng: phương pháp phân lớp cho phép người đọc có thể nhận diện được nhiều thành phần khác nhau và mối quan hệ giữa chúng.

Phương pháp biểu diễn cấu trúc theo các module cấu trúc chính để phân lớp kiến trúc của toàn bộ dự án nhúng: bởi vì trong hệ thống có rất nhiều module và các module này hoạt động độc lập với nhau, đồng thời chúng có những mối liên kết mức độ cao, do vậy phân lớp những loại module này làm tăng khả năng thể hiện cấu trúc hệ thống và không làm phức tạp khó hiểu cho người đọc.

Câu hỏi cuối chương

1. Thế nào là hệ thống nhúng, nêu những lĩnh vực ứng dụng của hệ thống nhúng
2. Phân biệt hai loại hệ thống thời gian thực
3. Trình bày về mô hình phát triển waterfall và spiral
4. Nêu những ưu điểm của phương pháp phân lớp trong biểu diễn kiến trúc hệ thống nhúng

Chương 2 - HỆ THỐNG PHẦN CỨNG

2.1 Tổng quan

Thông thường đi kèm với các hệ thống nhúng có rất nhiều những yêu cầu, thiết bị và bộ phận. Trong phần này chúng ta sẽ tìm hiểu một số bộ phận chính và một số yêu cầu chính của các hệ thống nhúng.

Bộ tính toán, xử lý

Đây là một trong những yêu cầu căn bản nhất của hệ thống nhúng. Tất cả các hệ thống lấy dữ liệu từ người dùng hoặc từ môi trường xung quanh. Công việc xử lý có thể được tiến hành bằng cách sử dụng bộ vi xử lý hoặc các vi mạch điện tử, các mạch động học. Phạm vi của giáo trình này là những hệ thống nhúng sử dụng vi xử lý với những bộ phận phần cứng khác.

Bộ tính toán được dùng để xử lý, biến đổi các dữ liệu thay đổi của người sử dụng hoặc môi trường thành những dữ liệu đầu ra theo yêu cầu.

Bộ phận xử lý tính toán logic này thường được tích hợp trong một chip hoặc một mạch điện tử. Khả năng xử lý của chúng được phát triển rất nhanh chóng và ngay bản thân chúng đã cung cấp cho ta những chức năng phức tạp. Lập trình viên sẽ phải khéo léo kết hợp những điểm mạnh này với chương trình của anh ta trong khi cài đặt một yêu cầu cụ thể.

Hệ thống nhúng cũng có thể lấy dữ liệu đầu vào từ môi trường xung quanh. Ví dụ, hệ thống âm thanh với các lựa chọn cài đặt như giả lập nhà hát, hội trường, rock vv. Người dùng có thể chuyển đổi các lựa chọn này theo yêu cầu của họ. Trong trường hợp này dữ liệu đầu vào là từ người sử dụng.

2.1.1 Bộ nhớ

Bộ nhớ là một trong những yêu cầu tài nguyên hiển nhiên không chỉ của hệ thống nhúng mà mọi hệ thống trên thực tế. Ngay cả hệ thống con người cũng cần bộ nhớ !

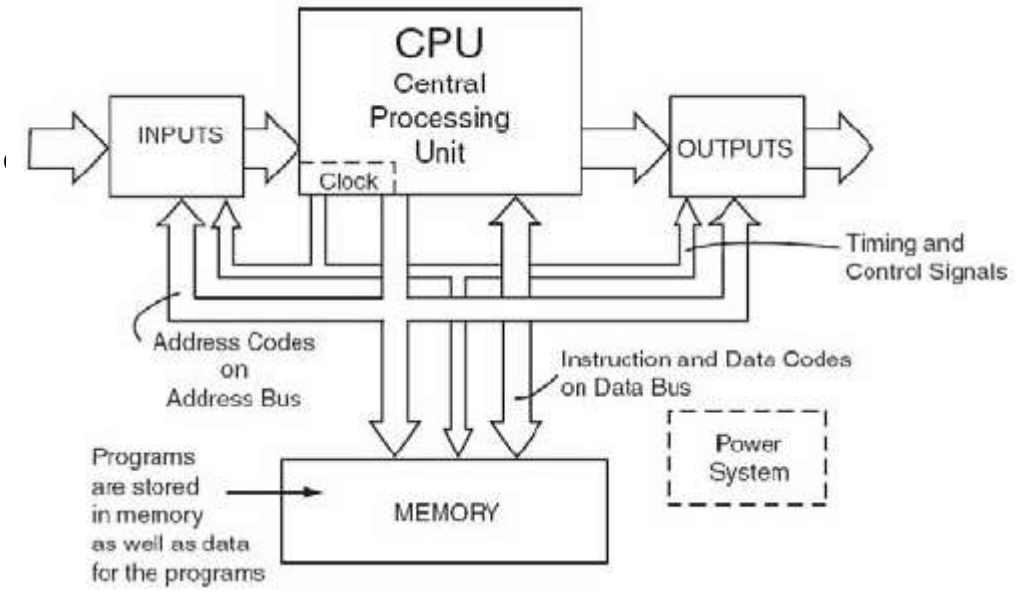
Thực tế cho thấy là bộ nhớ ngày càng rẻ hơn về giá cả và tốt hơn về chất lượng. Trong những thời gian trước, khi mỗi kilo byte, thậm chí là byte bộ nhớ đều được tính bằng tiền trăm ngàn thì mọi thao tác tác động vào bộ nhớ đều được tiến hành vô cùng chi tiết và cẩn thận. Với các hệ thống nhúng ta không có các khe cắm mở rộng, và bộ nhớ dành cho hệ thống nhúng thông thường vẫn là rất nhỏ. Vì vậy, bộ nhớ phải được xử lý có kế hoạch, chi tiết.

Những hạn chế về bộ nhớ như trên được thể hiện rõ ràng khi ta xem xét thiết kế của các hệ thống nhúng tiên tiến và các phần mềm đi kèm với chúng. Những thuật toán sử dụng nhiều bộ nhớ, sử dụng các cấu trúc dữ liệu cồng kềnh đều không được cài đặt trừ khi chúng thực sự cần thiết.

2.1.2 Các thành phần kết nối

Các thiết bị nhúng và ứng dụng không thể tồn tại độc lập, chúng cần có khả năng giao tiếp với các thiết bị khác để có thể thực hiện chức năng. Ta không thể yêu cầu người sử dụng kết nối thiết bị vào các khe cắm như Ethernet, vv... Những liên kết như vậy thường sử dụng các giao thức không dây như Bluetooth, WLAN, HiperLAN cho những khoảng cách ngắn hay 2.5G, 3G, 4G cho những khoảng cách xa hơn. Những thành phần kết nối khiến cho thiết bị trở nên thông minh hơn.

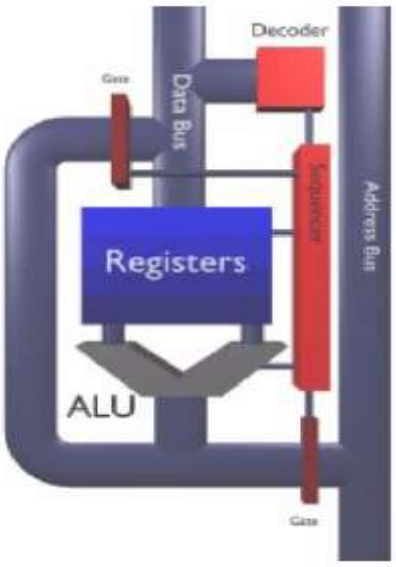
Các thành phần của hệ thống nhúng được minh họa trong hình vẽ dưới đây



Hình 2.1 Các thành phần phần cứng

2.2 Hệ vi xử lý

2.2.1 Tổng quan



Hình 2.2 Mô hình vi xử lý

Người ta vẫn biết tới phần lõi xử lý của các bộ VXL là đơn vị xử lý trung tâm CPU (Central Processing Unit) đóng vai trò như bộ não chịu trách nhiệm thực thi các phép tính và thực hiện các lệnh. Phần chính của CPU đảm nhiệm chức năng này là đơn vị

logic toán học (ALU – Arthimetic Logic Unit). Ngoài ra để hỗ trợ cho hoạt động của ALU còn có thêm một số các thành phần khác như bộ giải mã (decoder), bộ tuần tự (sequencer) và các thanh ghi.

Bộ giải mã chuyển đổi (thông dịch) các lệnh lưu trữ ở trong bộ mã chương trình thành các mã mà ALU có thể hiểu được và thực thi. Bộ tuần tự có nhiệm vụ quản lý dòng dữ liệu trao đổi qua bus dữ liệu của VXL. Các thanh ghi được sử dụng để CPU lưu trữ tạm thời các dữ liệu chính cho việc thực thi các lệnh và chúng có thể thay đổi nội dung trong quá trình hoạt động của ALU. Hầu hết các thanh ghi của VXL đều là các bộ nhớ được tham chiếu (mapped) và hội nhập với khu vực bộ nhớ và có thể được sử dụng như bất kỳ khu vực nhớ khác.

Các thanh ghi có chức năng lưu trữ trạng thái của CPU. Nếu các nội dung của bộ nhớ VXL và các nội dung của các thanh ghi tại một thời điểm nào đó được lữu giữ đầy đủ thì hoàn toàn có thể tạm dừng thực hiện phần chương trình hiện tại trong một khoảng thời gian bất kỳ và có thể trở lại trạng thái của CPU trước đó. Thực tế số lượng các thanh ghi và tên gọi của chúng cũng khác nhau trong các họ VXL/VĐK và thường do chính các nhà chế tạo qui định, nhưng về cơ bản chúng đều có chung các chức năng như đã nêu.

Khi thứ tự byte trong bộ nhớ đã được xác định thì người thiết kế phần cứng phải thực hiện một số quyết định xem CPU sẽ lưu dữ liệu đó như thế nào. Cơ chế này cũng khác nhau tuỳ theo kiến trúc tập lệnh được áp dụng. Có ba loại hình cơ bản:

- Kiến trúc ngăn xếp
- Kiến trúc bộ tích luỹ
- Kiến trúc thanh ghi mục đích chung

Kiến trúc ngăn xếp: sử dụng ngăn xếp để thực hiện lệnh và các toán tử nhận được từ đỉnh ngăn xếp. Mặc dù cơ chế này hỗ trợ mật độ mã tốt và mô hình đơn giản cho việc đánh giá cách thể hiện chương trình nhưng ngăn xếp không thể hỗ trợ khả năng truy

nhập ngẫu nhiên và hạn chế hiệu suất thực hiện lệnh.

Kiến trúc bộ tích luỹ: với lệnh một toán tử ngầm mặc định chứa trong thanh ghi tích luỹ có thể giảm được độ phức tạp bên trong của cấu trúc CPU và cho phép cấu thành lệnh rất nhỏ gọn. Nhưng thanh ghi tích luỹ chỉ là nơi chứa dữ liệu tạm thời nên giao thông bộ nhớ rất lớn.

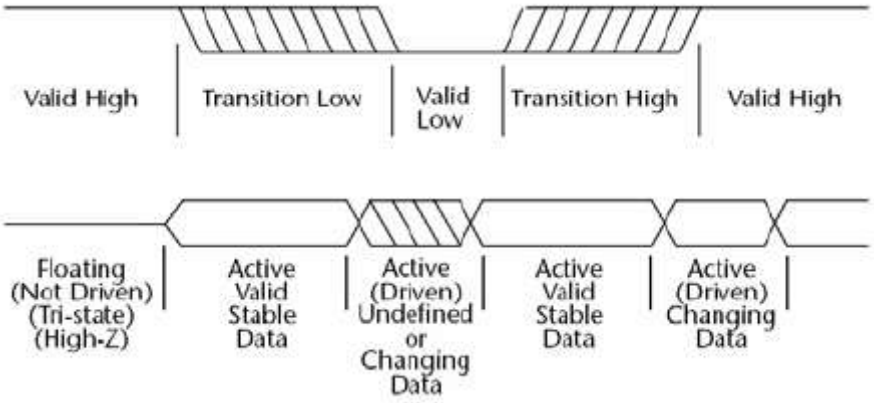
Kiến trúc thanh ghi mục đích chung: sử dụng các tập thanh ghi mục đích chung và được đón nhận như mô hình của các hệ thống CPU mới, hiện đại. Các tập thanh ghi đó nhanh hơn bộ nhớ thường và dễ dàng cho bộ biên dịch xử lý thực thi và có thể được sử dụng một cách hiệu quả. Hơn nữa giá thành phần cứng ngày càng có xu thế giảm đáng kể và tập thanh ghi có thể tăng nhanh. Nếu cơ chế truy nhập bộ nhớ nhanh thì kiến trúc dựa trên ngăn xếp có thể là sự lựa chọn lý tưởng; còn nếu truy nhập bộ nhớ chậm thì kiến trúc thanh ghi sẽ là sự lựa chọn phù hợp nhất.

Một số thanh ghi với chức năng điển hình thường được sử dụng trong các kiến trúc CPU như sau:

- **Thanh ghi con trỏ ngăn xếp (stack pointer)**: Thanh ghi này lưu giữ địa chỉ tiếp theo của ngăn xếp. Theo nguyên lý giá trị địa chỉ chứa trong thanh ghi con trỏ ngăn xếp sẽ giảm nếu dữ liệu được lưu thêm vào ngăn xếp và sẽ tăng khi dữ liệu được lấy ra khỏi ngăn xếp.
- **Thanh ghi chỉ số (index register)**: Thanh ghi chỉ số được sử dụng để lưu địa chỉ khi mode địa chỉ được sử dụng. Nó còn được biết tới với tên gọi là thanh ghi con trỏ hay thanh ghi lựa chọn tệp (Microchip).
- **Thanh ghi địa chỉ lệnh /Bộ đếm chương trình (Program Counter)**: Một trong những thanh ghi quan trọng nhất của CPU là thanh ghi bộ đếm chương trình. Thanh ghi bộ đếm chương trình lưu địa chỉ lệnh tiếp theo của chương trình sẽ được CPU xử lý. Mỗi khi lệnh được trỏ tới và được CPU xử lý thì nội dung giá trị của thanh ghi bộ đếm chương trình sẽ tăng lên một. Chương trình sẽ kết thúc khi thanh ghi PC có giá trị bằng địa chỉ cuối cùng của chương trình nằm trong bộ nhớ chương trình.
- **Thanh ghi tích lũy (Accumulator):** Thanh ghi tích lũy là một thanh ghi giao tiếp trực tiếp với ALU, được sử dụng để lưu giữ các toán tử hoặc kết quả của một phép toán trong quá trình hoạt động của ALU.

Xung nhịp và trạng thái tín hiệu

Trong VXL và các vi mạch số nói chung, hoạt động của hệ thống được thực hiện đồng bộ hoặc dị bộ theo các xung nhịp chuẩn. Các nhịp đó được lấy trực tiếp hoặc gián tiếp từ một nguồn xung chuẩn thường là các mạch tạo xung hoặc dao động thạch anh. Để mô tả hoạt động của hệ thống, các tín hiệu dữ liệu và điều khiển thường được mô tả trạng thái theo giản đồ thời gian và mức tín hiệu như được chỉ ra trong hình dưới: Mô tả và trạng thái tín hiệu hoạt động trong VXL



Hình 2.3 Mô tả và trạng thái tín hiệu hoạt động trong VXL

Mục đích của việc mô tả trạng thái tín hiệu theo giản đồ thời gian và mức tín hiệu là để phân tích và xác định chuỗi sự kiện hoạt động chi tiết trong mỗi chu kỳ bus. Nhờ việc mô tả này chúng ta có thể xem xét đến khả năng đáp ứng thời gian của các sự kiện thực thi trong hệ thống và thời gian cần thiết để thực thi hoạt động tuần tự cũng như là khả năng tương thích khi có sự hoạt động phối hợp giữa các thiết bị ghép nối hay mở rộng trong hệ thống. Thông thường thông tin về các nhịp thời gian hoạt động cũng như đặc tính kỹ thuật chi tiết được cung cấp hoặc qui định bởi các nhà chế tạo.

Một số đặc trưng về thời gian của các trạng thái hoạt động cơ bản của các tín hiệu hệ thống gồm có như sau:

- Thời gian tăng hoặc giảm
- Thời gian trễ lan truyền tín hiệu
- Thời gian thiết lập
- Thời gian giữ
- Trễ cấm hoạt động và trạng thái treo (Tri-State)
- Độ rộng xung
- Tần số nhịp xung hoạt động

2.2.2 Kiến trúc vi xử lý

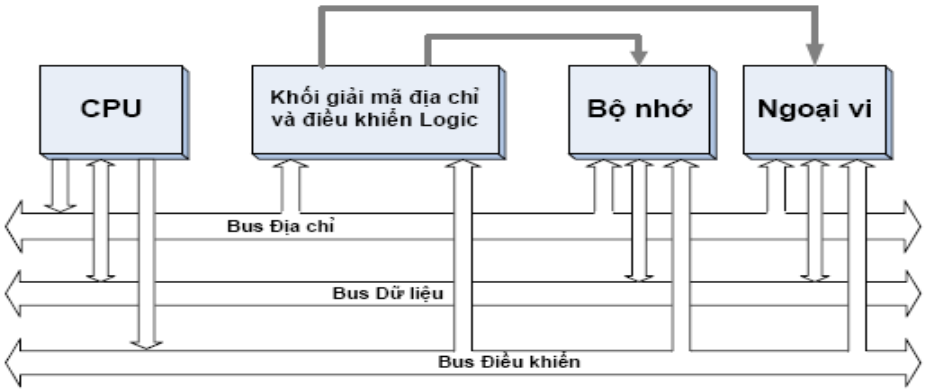
Trong phần này giới thiệu ngắn gọn cấu trúc nguyên lý của các chip xử lý nhúng ứng dụng trong các nền phần cứng nhúng hiện nay.

Sự phát triển nhanh chóng các chủng loại Chip khả trình với mật độ tích hợp cao đã và đang có một tác động đáng kể đến sự thay đổi trong việc thiết kế các nền phần cứng

thiết bị xử lý và điều khiển số trong thập kỷ gần đây. Mỗi chủng loại đều có những đặc điểm và phạm vi đối tượng ứng dụng và luôn không ngừng phát triển để đáp ứng một cách tốt nhất cho các yêu cầu công nghệ. Chúng đang hướng tới tập trung cho một thị trường công nghệ tiềm năng rộng lớn đó là các thiết bị xử lý và điều khiển nhúng. Trong bài viết này tác giả giới thiệu ngắn gọn về các chủng loại chip xử lý, điều khiển nhúng điển hình đang tồn tại và phát triển về một số đặc điểm và hướng phạm vi ứng dụng của chúng.

Có thể kể ra hàng loạt các Chíp khả trình có thể sử dụng cho các bài toán thiết kế hệ nhúng như các họ vi xử lý/vi điều khiển nhúng (Microprocessor/ Microcontroller), Chip DSP (Digital Signal Processing), các Chip khả trình trường (FPD – Field Programmable Device). Chúng ta dễ bị choáng ngợp nếu bắt đầu công việc thiết kế bằng việc tìm kiếm một Chip xử lý điều khiển phù hợp cho ứng dụng. Vì vậy cần phải có một hiểu biết và sự phân biệt về đặc điểm và ứng dụng của chúng khi lựa chọn và thiết kế. Các thông tin liên quan như nhà sản xuất cung cấp Chip, các kiến thức và công cụ phát triển kèm theo.

Chip vi xử lý/vi điều khiển nhúng là một chủng loại rất điển hình và đang được sử dụng rất phổ biến hiện này. Chúng được ra đời và sử dụng theo sự phát triển của các Chip xử lý ứng dụng cho máy tính. Vì đối tượng ứng dụng là các thiết bị nhúng nên cấu trúc cũng được thay đổi theo để đáp ứng các ứng dụng. Hiện nay chúng ta có thể thấy các họ vi xử lý điều khiển của rất nhiều các nhà chế tạo cung cấp như, Intel, Atmel, Motorola, Infineon. Về cấu trúc, chúng cũng tương tự như các Chíp xử lý phát triển cho PC nhưng ở mức độ đơn giản hơn nhiều về công năng và tài nguyên. Phổ biến vẫn là các Chip có độ rộng bus dữ liệu là 8bit, 16bit, 32bit. Về bản chất cấu trúc, Chip vi điều khiển là chip vi xử lý được tích hợp thêm các ngoại vi. Các ngoại vi thường là các khối chức năng ngoại vi thông dụng như bộ định thời gian, bộ đếm, bộ chuyển đổi A/D, giao diện song song, nối tiếp…Mức độ tích hợp ngoại vi cũng khác nhau tuỳ thuộc vào mục đích ứng dụng sẽ có thể tìm được Chip phù hợp. Thực tế với các ứng dụng yêu cầu độ tích hợp cao thì sẽ sử dụng giải pháp tích hợp trên chip, nếu không thì hầu hết các Chip đều cung cấp giải pháp để mở rộng ngoại vi đáp ứng cho một số lượng ứng dụng rộng và mềm dẻo.



Hình 2.4 Kiến trúc nguyên lý của VĐK với cấu trúc Havard

2.2.3 Sự thực thi

Trong phần này ta sẽ tìm hiểu về hoạt động của một số vi điều khiển sử dụng trong các hệ thống nhúng

Chip DSP

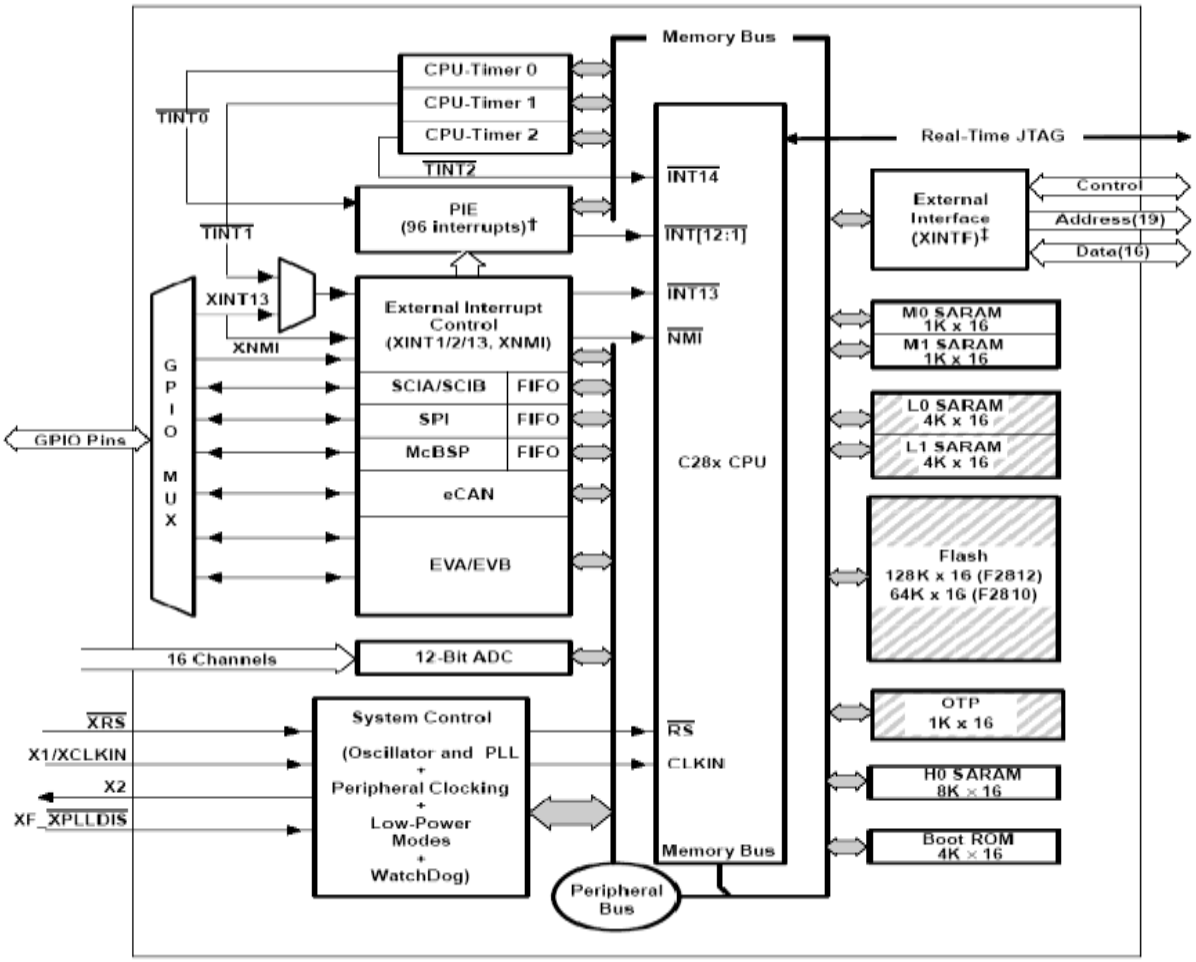
DSP vẫn được biết tới như một loại vi điều khiển đặc biệt với khả năng xử lý nhanh để phục vụ các bài toán yêu cầu khối lượng và tốc độ xử lý tính toán lớn. Với ưu điểm nổi bật về độ rộng băng thông của bus và thanh ghi tích luỹ, cho phép ALU xử lý song song với tốc độ đọc và xử lý lệnh nhanh hơn các loại vi điều khiển thông thường. Chip DSP cho phép thực hiện nhiều lệnh trong một nhịp nhờ vào kiến trúc bộ nhớ Havard.

Thông thường khi phải sử dụng DSP tức là để đáp ứng các bài toán tính toán lớn và tốc độ cao vì vậy định dạng biểu diễn toán học sẽ là một yếu tố quan trọng để phân loại và được quan tâm. Hiện nay chủ yếu chúng vẫn được phân loại theo hai kiểu là dấu phảy động và dấu phảy tĩnh. Đây cũng chính là một yếu tố quan trọng phải quan tâm đối với người thiết kế để lựa chọn được một DSP phù hợp với ứng dụng của mình. Các loại DSP dấu phảy tĩnh thường là loại 16bit hoặc 24bit còn các loại dấu phảy tĩnh thường là 32bit. Một ví dụ điển

hình về một DSP 16bit dấu phảy tĩnh là TMS320C55x, lưu các số nguyên 16-bit hoặc các số thực trong một miền giá trị cố định. Tuy nhiên các giá trị và hệ số trung gian có thể được lưu trữ với độ chính xác là 32bit trong thanh ghi tích luỹ 40bit nhằm giảm thiểu lỗi tính toán do phép làm tròn trong quá trình tính toán. Thông thường các loại DSP dấu phảy tĩnh có giá thành rẻ hơn các loại DSP dấu phảy động vì yêu cầu số lượng chân On-chip ít hơn và cần sử dụng lượng silicon ít hơn.

Ưu điểm nổi bật của các DSP dấu phảy động là có thể xử lý và biểu diễn số trong dải phạm vi giá trị rộng và động. Do đó vấn đề về chuyển đổi và hạn chế về phạm vi biểu diễn số không phải quan tâm như đối với loại DSP dấu phảy tĩnh. Một loại DSP 32bit dấu phảy tĩnh điển hình là TMS320C67x có thể xử lý và biểu diễn số gồm 24bit mantissa và 8bit exponent. Phần mantissa biểu diễn phần số lẻ trong phạm vi 1.0 – +1.0 và phần exponent biểu diễn vị trí của dấu phảy nhị phân và có thể dịch chuyển sang trái hoặc phải tuỳ theo giá trị số mà nó biểu diễn. Điều này trái ngược với các thiết kế trên nền DSP dấu phảy tĩnh, người phát triển chương trình phải tự qui ước, tính toán và phân chia ấn định thang biểu diễn số và phải luôn lưu tâm tới khả năng tràn số có thể xảy ra trong quá trình xử lý tính toán. Chính điều này đã gây ra khó khăn không nhỏ đối với người lập trình.

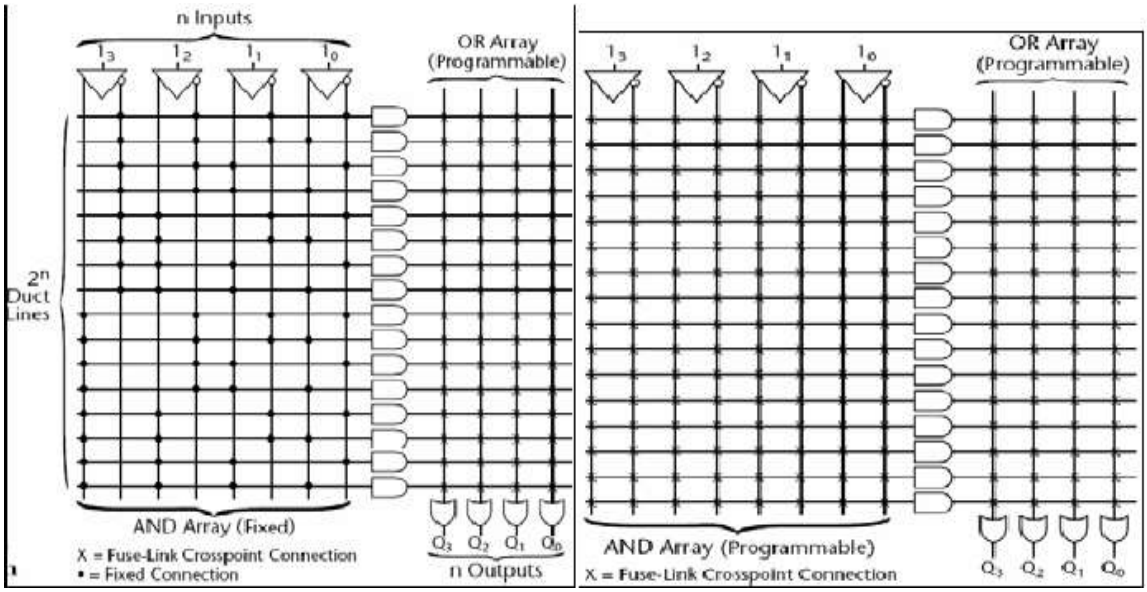
Những nhà thiết kế hệ thống phải quyết định vùng và độ chính xác cần thiết cho các ứng dụng. Các vi xử lý dấu phảy động thường được sử dụng cho các ứng dụng yêu cầu về độ chính xác cao và dải biểu diễn số lớn phù hợp với hệ thống có cấu trúc bộ nhớ lớn. Hơn nữa các DSP dấu phảy động cho phép phát triển phần mềm hiệu quả và đơn giản hơn bằng các trình biên dịch ngôn ngữ bậc cao như C do đó có thể giảm được giá thành và thời gian phát triển. Tuy nhiên giá thành lại cao nên các DSP dấu phảy động phù hợp với các ứng dụng khá đặc biệt và thường là với số lượng ít.



Hình 2.5: Giản đồ khối chức năng của DSP TMS320C28xx

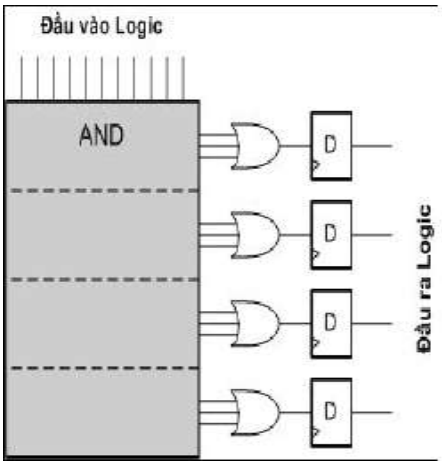
Chip PAL

Ngày nay khi nói đến các chủng loại Chip khả trình mảng ta thường biết tới một số tên gọi như PAL, CPLD, FPGA…Một chút lược sử về sự ra đời và phát triển sau đây sẽ giúp chúng ta hình dung được đặc điểm và nguồn gốc ra đời của chúng.



Hình 2.6 Cấu trúc PROM và PLA

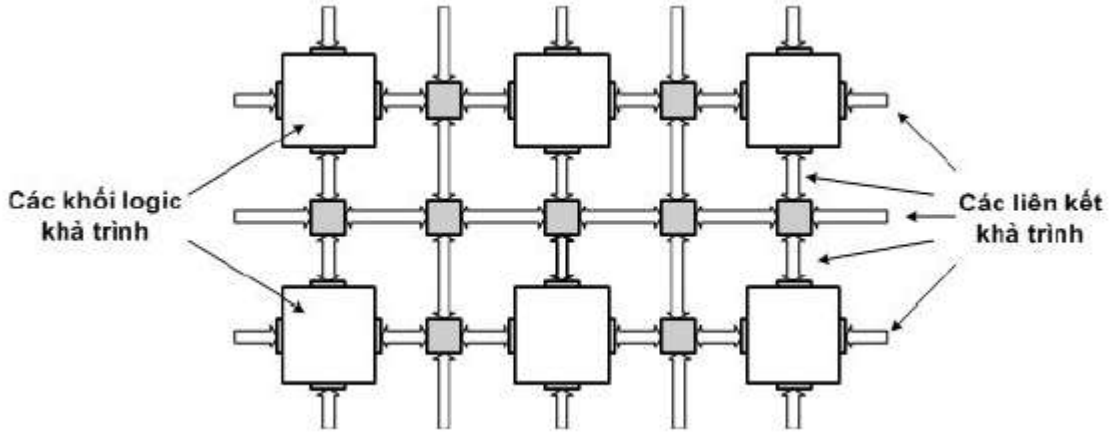
Lịch sử phát triển của các chủng loại Chip khả trình mảng PLA (Programmable Logic Array) được bắt nguồn từ nguyên lý bộ nhớ chương trình PROM (Programmable Read Only Memory). Trong đó các đầu vào địa chỉ đóng vai trò như các đường vào của mạch logic và các đường dữ liệu ra đóng vai trò như các đường ra của mạch logic. Vì PROM không thực sự phù hợp cho mục đích thiết kế các mạch logic nên PLA đã ra đời vào đầu thập kỷ 70. Nó rất phù hợp để thực hiện mạch logic có dạng tổng các tích (vì cấu thành bởi các phần tử logic AND và OR). Nhưng nhược điểm là chi phí sản xuất cao và tốc độ hoạt động thấp. Để khắc phục nhược điểm này PAL (Programmable Array Logic) đã được phát triển. Nó được cấu thành từ các phần tử AND khả trình và phần tử OR gán cố định và có chứa cả phần tử flipflop ở đầu ra nên có khả năng thực thi các mạch logic tuần tự. Hình 2.7 mô tả cấu trúc chung của PAL.



Hình 2.7 Cấu trúc chung của PAL

Từ khi được ra đời và phát triển PAL trở thành cơ sở cho sự ra đời của hàng loạt các chủng loại Chip khả trình mảng với cấu trúc phức tạp hơn như SPLD (Simple Programmable Logic Device), CPLD (Complex Programmable Logic Device), và sau này là FPGA (Field Programmable Gate Array). SPLD cũng là tên gọi cho nhóm các chủng loại Chip kiểu tương tự như PAL, PLA. Về mặt cấu trúc thì SPLD cho phép tích hợp logic với mật độ cao hơn so với PAL thông thường, nhưng kích thước của nó sẽ tăng lên rất nhanh nếu tiếp túc mở rộng và tăng mật độ tích hợp số đầu vào. Để đáp ứng nhu cầu mở rộng mật độ tích hợp CPLD đã được phát triển. Nó là sự tích hợp của nhiều khối SPLD và cung cấp thêm khả năng kết nối khả trình giữa các khối SPLD đơn lẻ với nhau. Với nguyên lý cấu trúc này CPLD có khả năng tích hợp với mật độ cao tương đương với 50 khối SPLD thông thường.

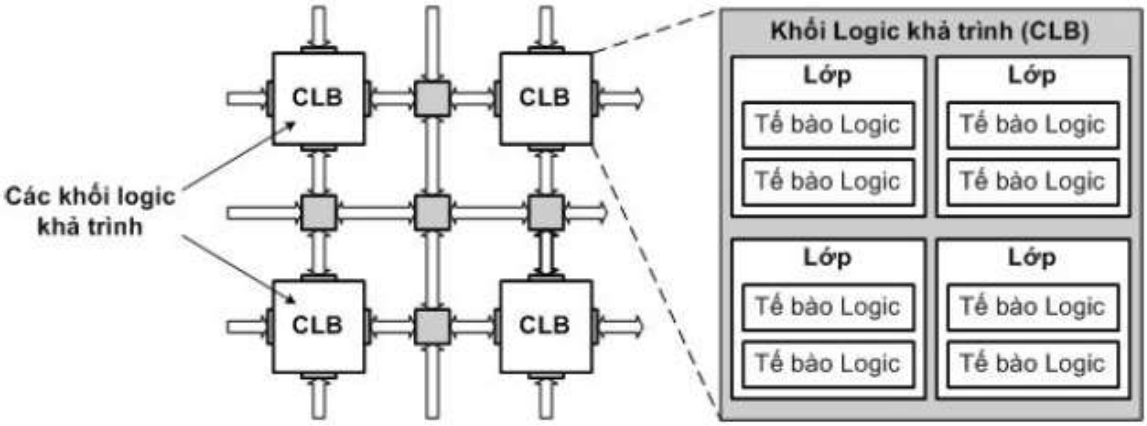
Nếu chỉ dừng đến đây chúng ta có thể thấy một đặc điểm chung của các chủng loại chip kiểu PLA hay CPLD đều cho phép thực hiện các mạch logic trên cơ sở tổ hợp logic của các đầu vào và ra bằng các phần tử AND và OR. Với nguyên lý này rõ ràng sẽ gặp khó khăn khi thực thi các ứng dụng đòi hỏi các phép tính toán logic phức tạp với tốc độ cao. Để đáp ứng điều này FPGA (Field Programmable Gate Arrays) đã ra đời. Nó là sự cấu thành của các khối logic khả trình cùng với các kênh kết nối liên thông khả trình giữa các khối đó với nhau. Một hình ảnh tiêu biểu về cấu trúc nguyên lý của FPGA được mô tả như trong Hình 2.8: Cấu trúc nguyên lý của FPGA.



Hình 2.8 Cấu trúc nguyên lý của FPGA

FPGA đang trở thành một sự lựa chọn thay thế rất cạnh tranh của các chip xử lý nhúng ASICs. Nó hỗ trợ các ưu điểm về chức năng lựa chọn giống như ASICs nhưng cho phép chỉnh sửa và thiết kế lại sau khi sử dụng và giá thành phát triển thấp hơn. FPGA cho phép khả năng thiết kế linh hoạt và thích nghi dễ dàng cho các tiện ích thiết bị tối ưu, trong khi vẫn duy trì được không gian kích thước phần cứng và năng lượng tiêu thụ của hệ thống. Điều này không dễ dàng nhận được khi thiết kế dựa trên nền các Chip DSP.

FPGA thực sự phù hợp cho các ứng dụng đòi hỏi lượng tính toán lớn như trong xử lý tín hiệu. FPGA có thể được lập trình hoạt động đồng thời với một số các đường dữ liệu song song. Chúng là các đường dữ liệu hoạt động của tổ hợp nhiều các chức năng từ đơn giản đến phức tạp như bộ cộng, bộ nhân, bộ đếm, bộ lưu trữ, bộ so sánh, bộ tính tương quan, …



Hình 2.9 Cấu trúc CLB và LAB

Ngày nay có thể phân loại ra một số kiểu chủng loại FPGA dựa vào cấu tạo của chúng:

Cấu tạo từ SRAM:Với loại này các mắt kết nối khả trình được thực hiện bằng các phần tử SRAM, chính vì vậy cho phép thực hiện lập trình lặp lại nhiều lần. Ưu điểm nổi bật của loại này là các ý tưởng thiết kế mới có thể được thực thi và thử nghiệm nhanh chóng. Hơn nữa SRAM cũng đang là một hướng phát triển rất mạnh hiện nay trong nền công nghiệp sản xuất bộ nhớ và cũng đều thực thi theo công nghệ CMOS rất phù hợp với công nghệ chế tạo FPGA.

Tuy nhiên một đặc điểm có thể xem như là nhược điểm của FPGA cấu tạo từ các phần tử SRAM là chúng phải cấu hình lại mỗi khi nguồn hệ thống được cung cấp. Công việc này thường được thực hiện bởi một bộ nhớ ngoài chuyên dụng hoặc bởi một bộ vi điều khiển kèm theo mạch. Chính vì vậy cũng làm giá thành của FPGA tăng thêm.

Cấu tạo từ cầu chì (antifused):

Không giống như loại FPGA cấu tạo từ SRAM, FPGA với cấu tử kiểu cầu chì được lập trình offline bằng một thiết bị lập trình chuyên dụng. Ý tưởng chế tạo loại FPGA này xuất phát từ nhu cầu về một thiết bị khả trình có khả năng lưu cấu hình sau khi được sử dụng. Tức là nó không phải làm công việc cấu hình mỗi khi nguồn hệ thống được cung cấp. Khi FPGA anti-fused đã được lập trình thì nó không thể bị thay đổi hay được lập trình lại nữa. Chính nhờ điều này nên nó không cần bất kỳ một bộ nhớ ngoài nào để lưu trữ cấu hình và có thể tiết kiệm, giảm giá thành của thiết bị.

Một ưu điểm nổi bật của FPGA antifused là kiểu cấu trúc liên kết khá bền vững với các loại nhiễu bức xạ. Đặc điểm này khá quan trọng khi thiết bị phải làm việc trong môi trường tiềm năng như quân sự hoặc hàng không vũ trụ. Vì vậy nó tránh được trường hợp rủi ro có thể xảy ra nếu sử dụng công nghệ SRAM là hiện tượng lật trạng thái (flipped). Tuy nhiên hiện tượng này cũng có thể được khắc phục bằng cơ chế dự phòng chập 3 nhưng lại làm tăng thêm chi phí chế tạo.

Một ưu điểm nổi bật của loại FPGA antifused là khả năng bảo vệ công nghệ. Tức là dữ liệu cấu hình lập trình cho FPGA có thể được bảo vệ bởi việc đọc bất hợp pháp hoặc không cho phép đọc. Trong qúa trình xử lý hoặc phát triển, người lập trình sẽ sử dụng một tệp dữ liệu cấu hình để lập trình và kiểm tra quá trình nạp cấu hình cho FPGA. Công việc này chỉ thực hiện một lần và sẽ không thể thay đổi được nữa. Khi thực hiện xong nó có thể được thiết lập thêm một thuộc tính là chống đọc trực tiếp từ FPGA dữ liệu liên quan đến cấu hình. Ngoài ra chúng ta cũng có thể biết thêm rằng FPGA antifused thường sử dụng ít năng lượng hơn loại

FPGA SRAM, kích thước cũng nhỏ hơn, và tốc độ cũng nhanh hơn một chút nhờ khoảng cách kết nối cứng giữa các phần tử ngắn hơn.

Tuy nhiên nhược điểm lớn nhất của FPGA anti-fused là chỉ có thể được lập trình và cấu hình một lần. Vì vậy nó chỉ thực sự phù hợp khi thực thi hoàn chỉnh sản phẩm cuối cùng và không phù hợp với mục đích thiết kế phát triển.

Cấu tạo từ EEPROM/FLASH

EEPROM or FLASH based FPGAs cũng có nguyên lý cấu tạo tương tự như loại FPGASRAM. Các phần tử cấu hình của nó được kết nối dựa trên một chuỗi thanh ghi dịch dài. Chúng có thể được cấu hình offline bằng các thiết bị lập trình chuyên dụng. Cũng có một số có thể lập trình online nhưng thời gian lập trình cấu hình sẽ gấp khoảng 3 lần thời gian thực thi với nền FPGA-SRAM. Khi đã được cấu hình đã được lập trình thì chúng có thể được duy trì và không bị mất đi như nguyên lý lưu giữ của EEPROM hoặc FLASH. Loại FPGA-EEPROM/FLASH có cấu tạo nhỏ hơn so với loại FPGA SRAM vì vậy cũng có thể giảm được thời gian lan truyền tín hiệu kết nối liên thông giữa các phần tử logic.

Để bảo vệ công nghệ khi FPGA đã được cấu hình và đưa ra sử dụng, ta có thể bảo vệ bằng cơ chế khóa mã mềm (cấu tạo từ khoảng 50 bit đến vài trăm bit). Muốn đọc được thông tin cấu hình trực tiếp từ FPGA, người ta cần phải có mã khóa đó và cũng rất khó hoặc không thể mò được theo nguyên lý thử sai. Vì muốn vậy theo ước tính cũng phải mất đến hàng triệu năm mới hy vọng thành công để mò ra được.

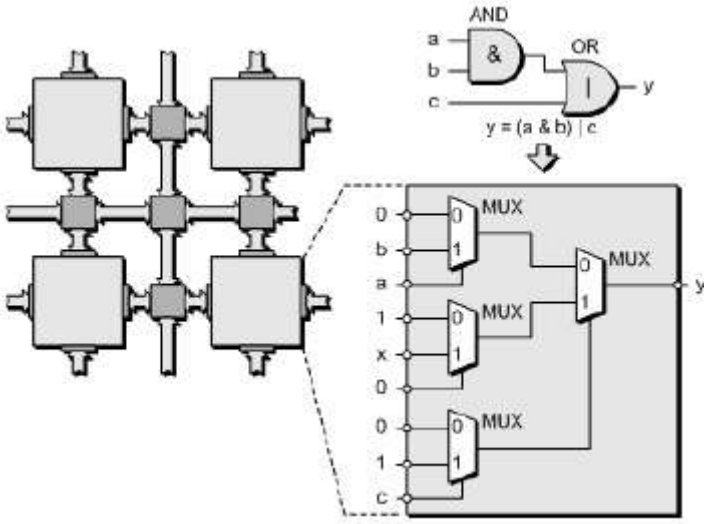
Tuy nhiên công nghệ chế tạo FPGA-EEPROM/FLASH đòi hỏi thực thi qua nhiều công đoạn xử lý hơn so với loại FPGA-SRAM vì vậy mà sự phát triển của chúng cũng chậm hơn. Hơn nữa năng lượng tiêu thụ của chúng cũng lớn hơn vì phải nuôi rất nhiều các phần tử điện trở kéo (pull-up resistor).

Cấu tạo từ tổ hợp FLASH/SRAM

Ngày nay người ta cũng phát triển chế tạo các loại FPGA cấu tạo từ các tổ hợp SRAM và FLASH để tận dụng được các ưu điểm của cả hai chủng loại này. Thông thường các phần tử cấu hình FLASH sẽ được sử dụng để lưu các nội dung cấu hình để sao chép cho các phần tử cấu hình SRAM. Và các phần tử cấu hình SRAM hoàn toàn có thể được cấu hình lại theo yêu cầu thiết kế trong khi vẫn duy trì một phần thiết kế cấu hình gốc lưu trong các phần tử FLASH.

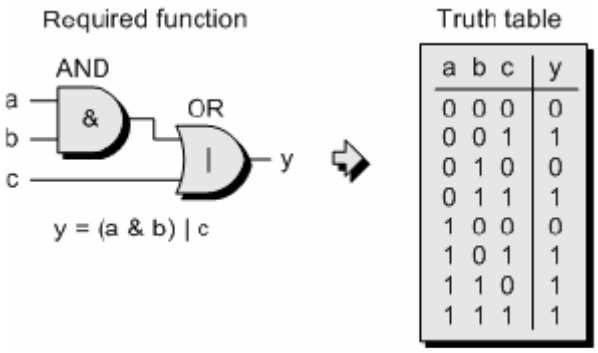
Người ta cũng thường phân loại FPGA dựa vào phần tử kiến trúc của chúng và bao gồm 3 loại chính: mịn, thô và trung bình. Bản chất việc phân loại này là dựa vào kiểu khối logic khả trình cấu thành nên FPGA. Với loại FPGA mịn thì kiến trúc các khối logic khả trình thường là các cổng logic đơn giản (kiểu AND, OR…, và các phần tử lưu giữ như Triger D…). Kiểu kiến trúc này phù hợp và thường sử dụng hiệu quả với kiến trúc ASICó. Gần đây xu thế phát triển của FPGA đang tập trung vào loại kiến trúc thô. Tức là các khối logic khả trình là các khối có khả năng xử lý logic lớn với nhiều tổ hợp liên kết và phức tạp với nhiều đầu vào và ra liên kết. Tuỳ theo mức độ của khối logic khả trình đó mà người ta phân ra thành các loại trung bình.

Có hai loại cấu trúc cơ bản cấu thành nên các khối logic khả trình trong kiến trúc FPGA thô hoặc trung bình là MUX (Multiplexer) và LUT (Lookup Table). Trong loại cấu trúc MUX thì các phần tử logic được cấu thành theo cấu trúc tổ hợp các đầu vào ra theo nguyên lý MUX như mô tả trong hình sau: Khối logic dạng MUX.



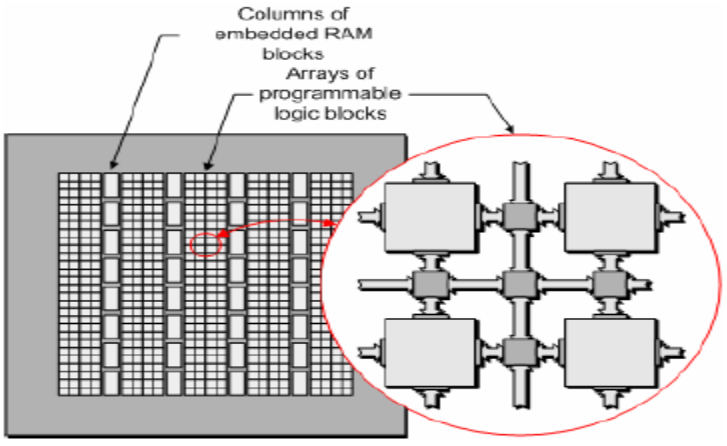
Hình 2.10 Khối logic dạng MUX

Đối với loại cấu trúc LUT thì các đầu vào thực chất là các tổ hợp để chọn ra giá trị trong bảng chất lý của hàm chức năng cần thực thi. Nguyên lý của loại khối logic này được mô tả như trong Hình 2.11.



Hình 2.11 LUT thực hiện hàm tổ hợp AND và OR

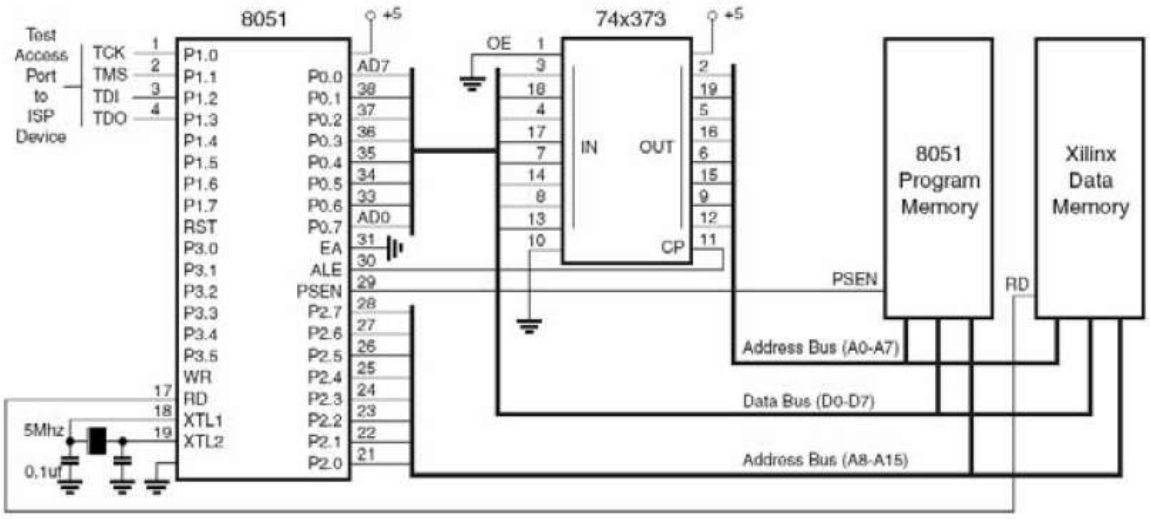
Hầu hết các ứng dụng đều có nhu cầu về bộ nhớ RAM on Chip vì vậy một số dòng FPGA hiện nay cũng tích hợp thêm cả các phần tử nhớ RAM và được gọi là RAM nhúng (embedded RAM). Các phần tử RAM đó được tổ chức thành từng khối và tuỳ thuộc vào kiến trúc của FPGA nó sẽ được phân bố linh hoạt, thường là xung quanh các phần tử ngoại vi hoặc phân bố đều trên bề mặt Chip. Một hình ảnh minh hoạ về phân bố RAM trong kiến trúc FPGA được mô tả như trong Hình 2.12.



Hình 2.12 Hình ảnh Chip có các cột là các khối RAM nhúng

FPGA với hạt nhân DSP

Thực chất đó là một tổ hợp nhằm tăng tốc và khả năng tính toán. Khái niệm này cũng tương tự như các bộ đồng xử lý toán học trong kiến trúc máy tính. Nguyên lý là nhằm san sẻ và giảm bớt tải sang FPGA để thực thi các chức năng tính toán lớn (thông thường đòi hỏi thực hiện trong nhiều nhịp hoạt động của Chip DSP) và cho phép Chip DSP tập trung thực hiện các chức năng đơn nhịp tối ưu. Tổ hợp FPGA và DSP là một kiến trúc rất linh hoạt và đặc biệt cải thiện được hiệu suất thực hiện và tăng tốc hơn rất nhiều so với kiến trúc nhiều Chip DPS hoặc ASICs đồng thời giá thành lại thấp hơn.



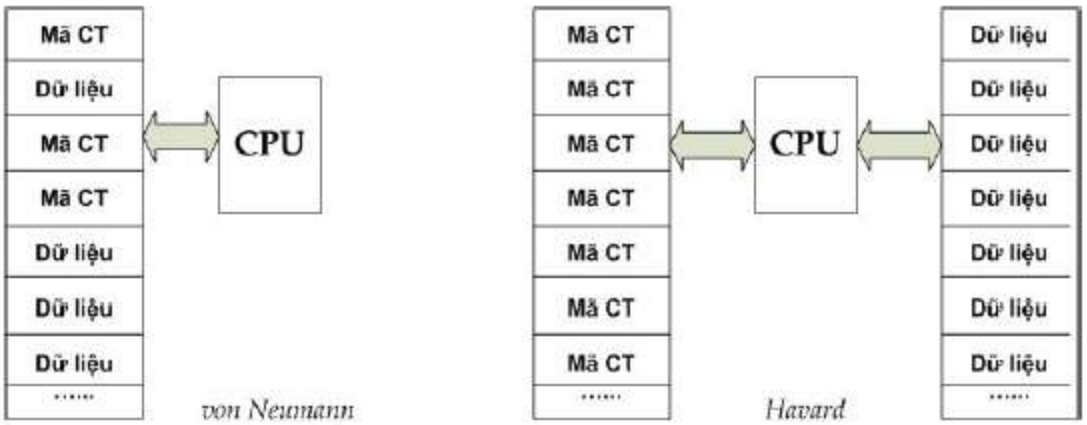
Hình 2.13 Sơ đồ nguyên lý mạch ghép nối VĐK và FPGA

2.3 Bộ nhớ

2.3.1 Tổng quan

Kiến trúc bộ nhớ được chia ra làm hai loại chính và được áp dụng rộng rãi trong hầu hết các Chip xử lý nhúng hiện nay là kiến trúc bộ nhớ von Neumann và Havard.

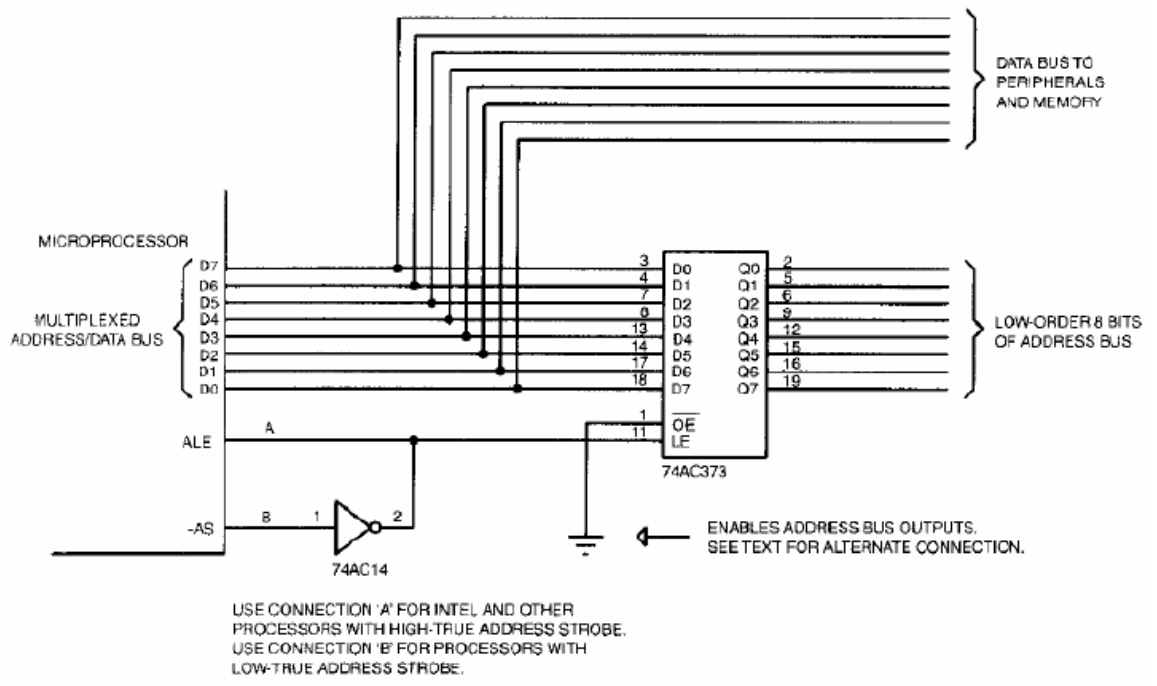
Trong kiến trúc von Neumann không phân biệt vùng chứa dữ liệu và mã chương trình. Cả chương trình và dữ liệu đều được truy nhập theo cùng một đường. Điều này cho phép đưa dữ liệu vào vùng mã chương trình ROM, và cũng có thể lưu mã chương trình vào vùng dữ liệu RAM và thực hiện từ đó.



Hình 2.14 Kiến trúc bộ nhớ von Neumann và Havard

Kiến trúc Havard tách/phân biệt vùng lưu mã chương trình và dữ liệu. Mã chương trình chỉ có thể được lưu và thực hiện trong vùng chứa ROM và dữ liệu cũng chỉ có thể lưu và trao đổi trong vùng RAM. Hầu hết các vi xử lý nhúng ngày nay sử dụng kiến trúc bộ nhớ Havard hoặc kiến trúc Havard mở rộng (tức là bộ nhớ chương trình và dữ liệu tách biệt nhưng vẫn cho phép khả năng hạn chế để lấy dữ liệu ra từ vùng mã chương trình). Trong kiến trúc bộ nhớ Havard mở rộng thường sử dụng một số lượng nhỏ các con trỏ để lấy dữ liệu từ vùng mã chương trình theo cách nhúng vào trong các lệnh tức thời. Một số Chip vi điều khiển nhúng tiêu biểu hiện nay sử dụng cấu trúc Havard là 8031, PIC, Atmel AVR90S. Nếu sử dụng Chip 8031 chúng ta sẽ nhận thấy điều này thông qua việc truy nhập lấy dữ liệu ra từ vùng dữ liệu RAM hoặc từ vùng mã chương trình. Chúng ta có một vài con trỏ được sử dụng để lấy dữ liệu ra từ bộ nhớ dữ liệu RAM, nhưng chỉ có duy nhất một con trỏ DPTR có thể được sử dụng để lấy dữ liệu ra từ vùng mã chương trình. Hình 2.14 mô tả nguyên lý kiến trúc của bộ nhớ von Neumann và Harvard.

Ưu điểm nổi bật của cấu trúc bộ nhớ Harvard so với kiến trúc von Neumann là có hai kênh tách biệt để truy nhập vào vùng bộ nhớ mã chương trình và dữ liệu nhờ vậy mà mã chương trình và dữ liệu có thể được truy nhập đồng thời và làm tăng tốc độ luồng trao đổi với bộ xử lý.



Hình 2.15 Nguyên lý điều khiển tách kênh truy nhập bus địa chỉ và bus dữ liệu

2.3.2 Bộ nhớ ROM

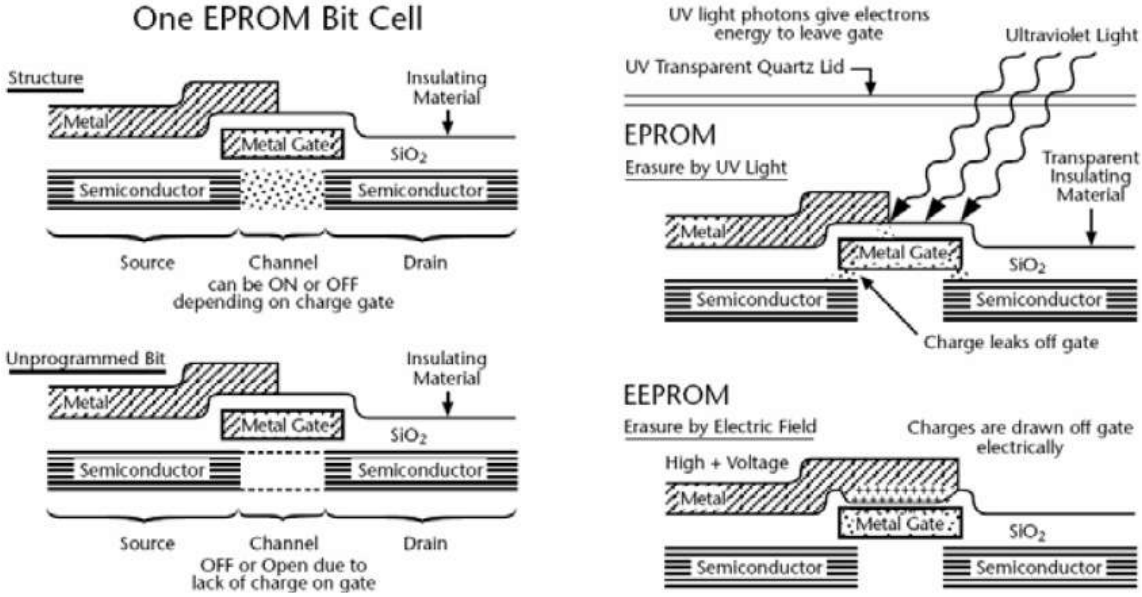
Bộ nhớ chỉ đọc ROM cũng được chế tạo bằng công nghệ bán dẫn. Chương trình trong ROM được viết vào lúc chế tạo nó. Thông thường, ROM chứa chương trình khởi động máy tính, chương trình điều khiển trong các thiết bị điều khiển tự động,...

PROM (Programable ROM): Chế tạo bằng các mối nối (cầu chì - có thể làm đứt bằng điện). Chương trình nằm trong PROM có thể được viết vào bởi người sử dụng bằng thiết bị đặc biệt và không thể xóa được.

EEPROM (Eletrically Erasable Programable ROM): Chế tạo bằng công nghệ bán dẫn. Chương trình nằm trong ROM có thể được viết vào và có thể xóa (bằng điện) để viết lại bởi người sử dụng.

EPROM (Erasable Programable ROM): Chế tạo bằng nguyên tắt phân cực tĩnh điện. Chương trình nằm trong ROM có thể được viết vào (bằng điện) và có thể xóa (bằng tia cực tím - trung hòa tĩnh điện) để viết lại bởi người sử dụng.

Bao gồm một mảng các transistor khả trình. Mã chương trình sẽ được ghi trực tiếp và vi xử lý có thể đọc ra để thực hiện. EPROM có thể xoá được bằng tia cực tím và có thể được lập trình lại. Cấu trúc vật lý của EPROM được mô tả như trong Hình 2.16.

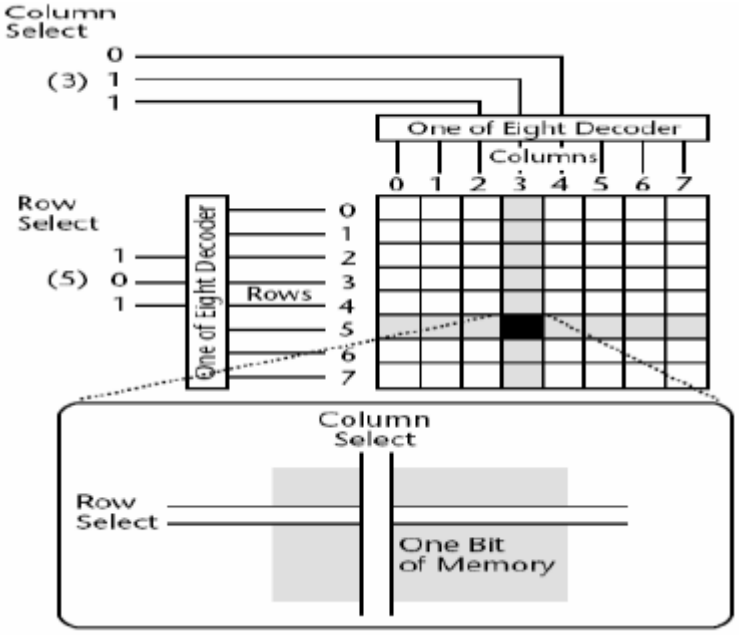


Hình 2.16 Nguyên lý cấu tạo và hoạt động xoá của EPROM

Bộ nhớ Flash: Cũng giống như EPROM được cấu tạo bởi một mảng transistor khả trình nhưng có thể xoá được bằng điện và chính vì vậy có thể nạp lại chương trình mà không cần tách ra khỏi nền phần cứng VXL. Ưu điểm của bộ nhớ flash là có thể lập trình trực tiếp trên mạch cứng mà nó đang thực thi trên đó.

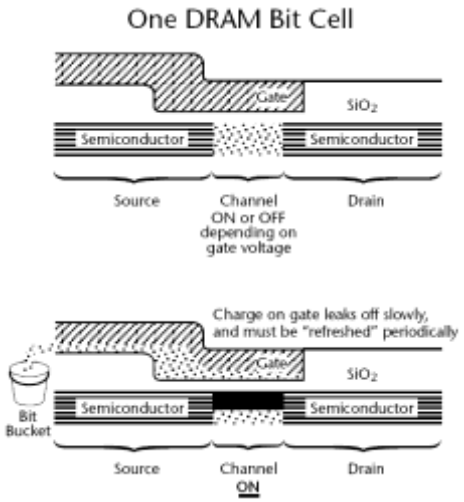
2.3.3 Bộ nhớ RAM

Vùng để lưu hoặc trao đổi dữ liệu trung gian trong quá trình thực hiện chương trình.

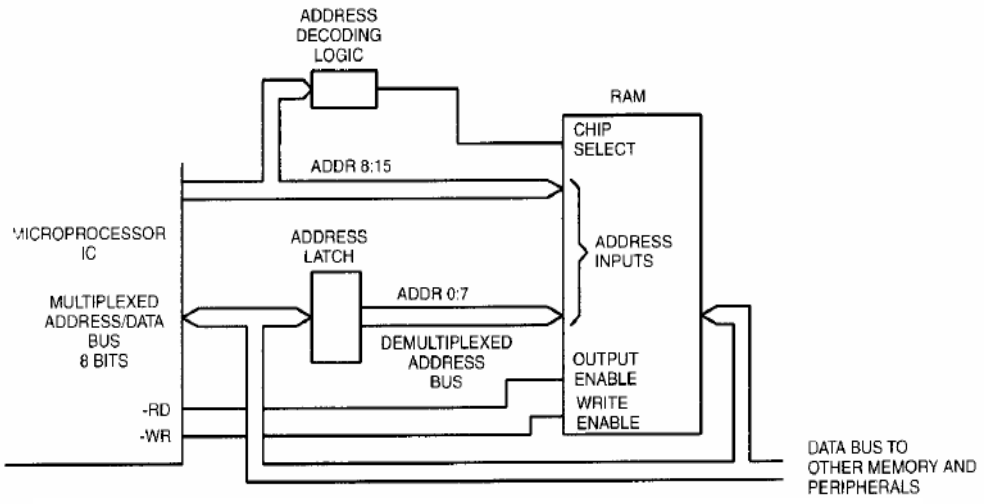


Hình 2.17 Cấu trúc nguyên lý bộ nhớ RAM

RAM có hai loại SRAM và DRAM



Hình 2.18 Cấu trúc một phần tử nhớ DRAM



Hình 2.19 Nguyên lý ghép nối (mở rộng) RAM với VXL

2.3.4 Quản lý bộ nhớ

Công việc quản lý bộ nhớ của máy vi tính chủ yếu là do bộ vi xử lý đảm nhiệm. Bên cạnh đó còn có DMAC (Direct Memory Access Controller) cũng tham gia quản lý bộ nhớ trong việc truyền số liệu giữa controller ổ đĩa với bộ nhớ và làm tươi bộ nhớ. ở những máy có Cache Memory thì Cache Memory Controller thực hiện các công việc truyền số liệu giữa Cache Memory và RAM.

Ở khu vực trung tâm của máy vi tính (bộ vi xử lý, ROM, RAM, các bus...), thực chất của việc quản lý bộ nhớ là các thanh ghi của vi xử lý đưa ra các địa chỉ của ô nhớ hoặc của cổng I/O qua bus địa chỉ, cùng các lệnh điều khiển/ trạng thái khác và lệnh đọc vào/ viết ra các số liệu của các ô nhớ ấy. Các bộ phận bên ngoài VXL sẽ giải mã các địa chỉ và các tín hiệu điều khiển/ trạng thái đó để trỏ vào các byte/ từ/ từ kép... của bộ nhớ để thực hiện các thao tác tương ứng.

Còn từ các ổ đĩa trở đi, việc quản lý bộ nhớ là thực hiện các lệnh của hệ điều hành lên các file (có địa chỉ 3 chiều là C-H-S), cụ thể là truyền số liệu nhờ DMAC giữa vùng đệm (buffer) của bộ điều khiển ổ đĩa với bộ nhớ RAM.

Các bộ vi xử lý Intel từ thế hệ 286 trở đi phân biệt hai mode địa chỉ: mode địa chỉ thực (chỉ quản lý 20 bit địa chỉ vật lý của bộ nhớ) và mode địa chỉ bảo vệ (quản lý tới 32 bit địa chỉ ảo nhờ các thanh ghi ẩn trong bộ vi xử lý).

Ở cấp dưới, tức cấp ngoại vi, như bộ điều khiển ổ đĩa, bộ điều khiển màn hình, máy in... cũng có tổ chức bộ nhớ riêng của chúng để tiện cho việc cất giữ và xử lý với các đặc thù riêng.

Các bộ nhớ RAM-ROM và các vùng nhớ của bộ nhớ ngoài (trên các ổ đĩa), khác nhau về cách mã hoá các bit, cách tổ chức, do đó cả cách truy nhập cũng khác nhau.

Bộ nhớ của vi xử lý có thể xem như bao gồm có bộ nhớ ROM và bộ nhớ RAM. Bộ nhớ RAM của vi xử lý chính là các thanh ghi (thanh ghi chung, thanh ghi chỉ số, thanh ghi đoạn, thanh ghi ngăn xếp, thanh ghi trạng thái, thanh ghi cờ, các bộ đệm số liệu/ địa chỉ/ điều khiển...). Còn bộ nhớ RAM là bộ phận giải mã lệnh để phát ra các vi lệnh.

Nhằm mục đích quản lý được số lượng địa chỉ nhớ (ảo) nhiều hơn số đường địa chỉ của bộ vi xử lý và bảo vệ các vùng nhớ của các nhiệm vụ khác nhau (task) và của hạt nhân (kernal) chống truy nhập không hợp pháp, các vi xử lý có các cách tổ chức đặc biệt các thanh ghi địa chỉ (bộ phận phân trang, điều khiển đoạn của các nhiệm vụ).

Các bộ vi xử lý từ thế hệ 486 trở đi còn có một bộ nhớ Cache Memory với kích thước nhiều Kbyte để chứa mảng các lệnh và số liệu đang thường dùng lấy từ bộ nhớ RAM, nhằm tăng tốc độ truy nhập.

Để tăng tốc độ tính toán các phép toán dấu chấm động, trong các bộ vi xử lý từ 486 trở đi còn có bộ phận dấu chấm động (FPU, Floating Point Unit), bộ phận này cũng có các thanh ghi FPU phục vụ riêng cho nó.

Bộ nhớ trong của máy tính dùng để chứa chương trình và số liệu của phần chương trình hạt nhân và các nhiệm vụ. Mỗi byte được gán cho một địa chỉ để VXL và DMAC có thể truy nhập tới.

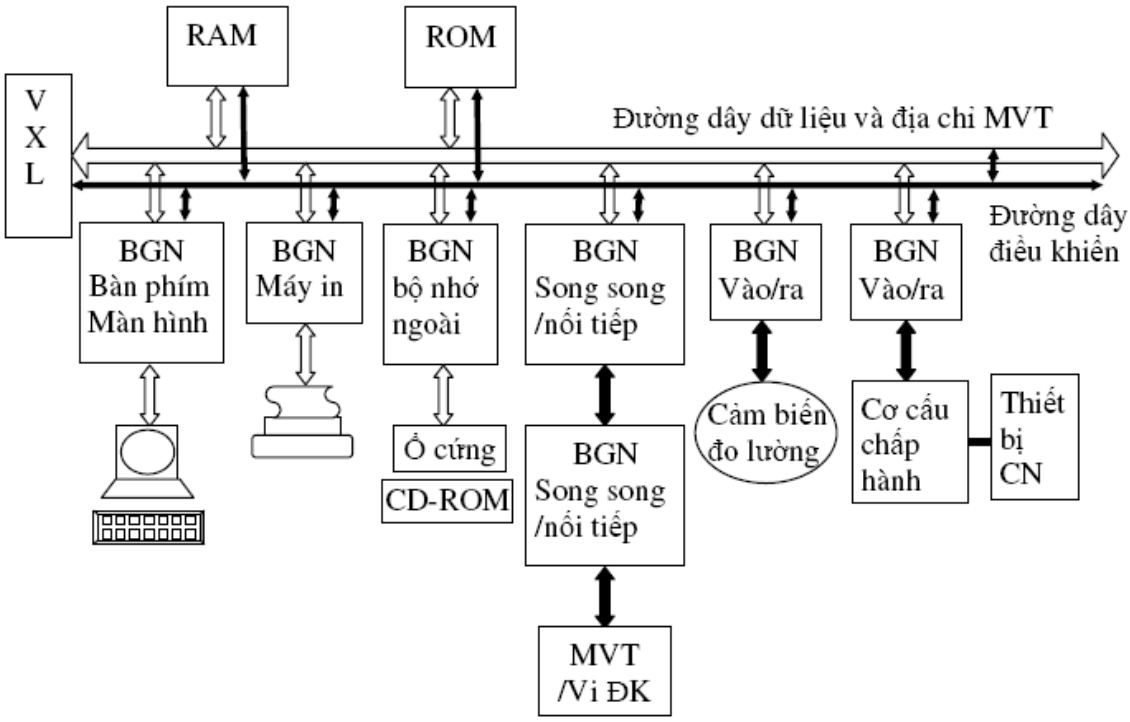
Bộ nhớ RAM ở những máy từ 386 trở đi có thể được tách riêng ra bộ nhớ đệm (cache memory), là RAM tĩnh với thời gian truy nhập nhanh, có kích thước dưới 1Mb được nối ngay vào bus nội bộ của máy tính sát ngay vi xử lý và được điều khiển bởi Cache controller. Phần còn lại là DRAM, chậm hơn nhưng rẻ hơn và có dung lượng lớn hơn.

2.4 Thiết bị ngoại vi

2.4.1 Tổng quan

Máy vi tính hay hệ vi xử lý đều có cấu trúc chung gồm khối xử lý trung tâm CPU, bộ nhớ và các mạch vào ra I/O. Ngoài ra, máy vi tính còn phải trao đổi dữ liệu với môi trường bên ngoài, ví dù giao tiếp với người sử dụng qua bàn phím, màn hình, trao đổi với các thiệt bị ngoài thông dùng, các thiệt bị trong hệ đo lường, điều khiển và các máy vi tính khác trong mạng. Các thiết bị ngoại vi bao gồm:

- Các thiết bị vào chuẩn như bàn phím, chuột
- Các thiết bị ra chuẩn như màn hình, máy in
- Các bộ nhớ ngoài chuẩn như ổ cứng, CD ROM
- Các hệ đo lường, điều khiển
- ...



Hình 2.20 Cấu trúc ghép nối giữa máy vi tính và thiết bị ngoại vi

Trong đó

- VXL : vi xử lý
- BGN: bộ ghép nối
- CN: công nghiệp
- ĐK: điều khiển

Bộ phối ghép nằm trung gian giữa máy vi tính và các thiết bị ngoài, đóng vai trò trung chuyển dữ liệu (nhận và truyền) giữa chúng.

Khi truyền dữ liệu từ máy vi tính ra thiết bị ngoài, bộ phối ghép đóng vai trò nhận dữ liệu từ máy tính và là nguồn cấp dữ liệu cho thiết bị ngoài.

Khi truyền dữ liệu từ thiết bị ngoài vào máy vi tính, bộ phối ghép đóng vai trò nhận dữ liệu từ thiết bị ngoài và là nguồn cấp dữ liệu vào cho máy tính.

Bộ phối ghép làm nhiệm vụ phối hợp trao đổi dữ liệu giữa máy tính và thiết bị ngoài về mức và công suất của tín hiệu, về dạng tín hiệu, về tốc độ và phương thức trao đổi.

Phối hợp về mức và công suất tín hiệu

Mức tín hiệu của máy vi tính thường là mức (0V, 5V) trong khi của các thiết bị ngoài, hoặc ở mức cao (± 15V, ± 48V) hoặc rất thấp (<<1V). Do đó, bộ phối ghép phải biến đổi các mức trên cho phù hợp.

Công suất của các tín hiệu trên bus dữ liệu của máy vi tính rất nhỏ (cỡ vài chục mA), trong khi cần công suất lớn hơn nhiều cho thiết bị ngoài. Do đó bộ phối ghép phải biến đổi công suất cho phù hợp.

Ở các ngõ vào và ngõ ra của bộ phối ghép thường dùng các mạch đệm ba trạng thái.

Phối hợp về dạng dữ liệu (tín hiệu).

Bộ phối ghép phải đảm bảo tính tương thích về cơ chế trao đổi dữ liệu giữa máy tính và thiết bị ngoài.

Phối hợp về tốc độ trao đổi dữ liệu.

Máy tính thường hoạt động với tốc độ cao, trong khi các thiết bị ngoài thường hoạt động chậm hơn. Do đó bộ phối ghép phải có khả năng cấp, nhận dữ liệu nhanh với máy tính, nhưng vói thiết bị ngoại thì ngược lại .

Phối hợp về phương thức trao đổi dữ liệu.

Để đảm bảo sự trao đổi dữ liệu một cách tin cậy, cần có bộ phối ghép và phương thức trao đổi dữ liệu diễn ra theo một trình tự nhất định và hợp lý. Nếu việc trao đổi dữ liệu do máy tính yêu cầu thì quá trình diễn ra như sau:

- Máy tính đưa lệnh điều khiển để khởi động bộ phối ghép hay thiết bị ngoài.
- Máy tính đọc tín hiệu trả lời. Nếu có tín hiệu sẵn sàng mới trao đổi tin, nếu không, thêm một chu kỳ chờ và đọc lại trạng thái.
- Máy tính trao đổi tin khi đọc thấy trạng thái sẵn sàng.

Nếu việc trao đổi tin do TBN yêu cầu: để giảm thời gian chờ đợi trạng thái sẵn sàng của TBN, máy tính có thể khởi động TBN rồi thực hiện các nhiệm vụ khác. Việc trao đổi tin diễn ra khi:

- TBN gửi yêu cầu trao đổi tin tới bộ xử lý ngắt của khối ghép nối, để đưa yêu cầu ngắt chương trình đến máy tính.
- Nếu có nhiều thiết bị ngoài cùng gửi yêu cầu, KGN xử lý theo mức ưu tiên ngắt định trước, rồi đưa yêu cầu trao đổi tin cho máy tính.
- Máy tính nhận yêu cầu, chuẩn bị trao đổi và gửi tín hiệu xác nhận sẵn sàng trao đổi.
- KGN nhận và truyền tín hiệu xác nhận cho TBN.
- TBN trao đổi tin với KGN và KGN trao đổi tin với máy tính (nếu là đưa tin vào) hoặc máy tính trao đổi tin với KGN và KGN trao đổi tin với TBN (nếu là đưa tin ra).

2.4.2 Vào ra nối tiếp

Cổng nối tiếp được sử dụng để truyền dữ liệu hai chiều giữa máy tính và ngoại vi, có các ưu điểm sau:

- Khoảng cách truyền xa hơn truyền song song.
- Số dây kết nối ít.
- Có thể truyền không dây dùng hồng ngoại.
- Có thể ghép nối với vi điều khiển hay PLC (Programmable Logic Device).
- Cho phép nối mạng.
- Có thể tháo lắp thiết bị trong lúc máy tính đang làm việc.
- Có thể cung cấp nguồn cho các mạch điện đơn giản

Các thiết bị ghép nối chia thành 2 loại: DTE (Data Terminal Equipment) và DCE (Data Communication Equipment). DCE là các thiết bị trung gian như MODEM còn DTE là các thiết bị tiếp nhận hay truyền dữ liệu như máy tính, PLC, vi điều khiển, … Việc trao đổi tín hiệu thông thường qua 2 chân RxD (nhận) và TxD (truyền). Các tín hiệu còn lại có chức năng hỗ trợ để thiết lập và điều khiển quá trình truyền, được gọi là các tín hiệu bắt tay (handshake). Ưu điểm của quá trình truyền dùng tín hiệu bắt tay là có thể kiểm soát đường truyền.

Tín hiệu truyền theo chuẩn RS-232 của EIA (Electronics Industry Associations). Chuẩn RS-232 quy định mức logic 1 ứng với điện áp từ -3V đến -25V (mark), mức logic 0 ứng với điện áp từ 3V đến 25V (space) và có khả năng cung cấp dòng từ 10 mA đến 20 mA. Ngoài ra, tất cả các ngõ ra đều có đặc tính chống chập mạch.

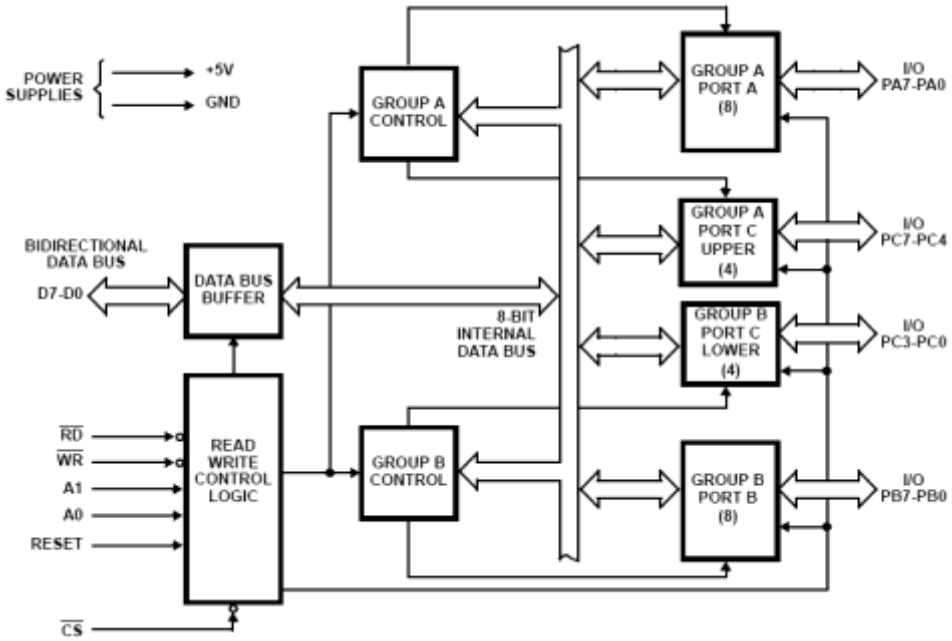
Chuẩn RS-232 cho phép truyền tín hiệu với tốc độ đến 20,000 bps nhưng nếu cáp truyền đủ ngắn có thể lên đến 115,200 bps. Các phương thức nối giữa DTE và DCE:

- Đơn công (simplex connection): dữ liệu chỉ được truyền theo 1 hướng.

- Bán song công (half-duplex): dữ liệu truyền theo 2 hướng, nhưng mỗi thời điểm chỉ được truyền theo 1 hướng.
- Song công (full-duplex): số liệu được truyền đồng thời theo 2 hướng.

2.4.3 Vào ra song song

Trong phần này ta xét cổng song song khả trình 82C55A. 82C55A là một giao diện ngoại vi cổng song song khả trình được chế tạo theo công nghệ CMOS. Nó là một thiết bị ngoại vi vào ra khả trình đa mục đích và có thể được sử dụng với nhiều loại VXL/VĐK khác nhau. 82C55A có 24 chân vào ra on Chip được chia ra thành 2 nhóm, mỗi nhóm 12 chân và có thể được sử dụng theo 3 chế độ hoạt động khác nhau. Hình 2.21 mô tả giản đồ khối chức năng của chip 82C55A.

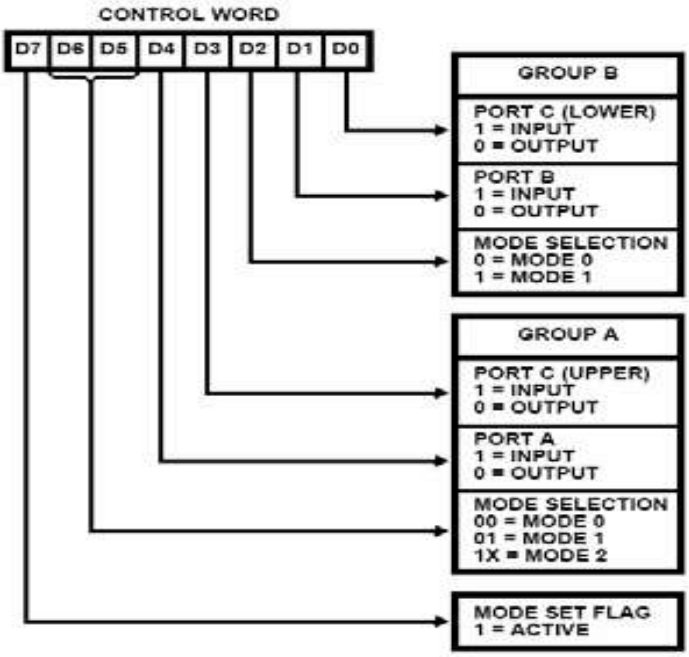


Hình 2.21 Giản đồ chức năng của 82C55A

82C55A cung cấp 3 chế độ hoạt động chính và có thể lập trình để lựa chọn

- Mode 0: Hoạt động vào ra cơ bản
- Mode 1: Hoạt động vào ra nắm bắt (strobed)
- Mode 2: Hoạt động Bus 2 chiều

Việc lựa chọn chế độ hoạt động được thực hiện thông qua thanh ghi từ điều khiển và được mô tả như trong hình sau.



Hình 2.22 Thanh ghi từ điều khiển chọn chế độ hoạt động cho 82C55A

Khi đầu vào RESET được điều khiển ở mức cao thì tất cả các cổng sẽ được thiết lập hoạt động ở chế độ cổng vào với 24 đường tín hiệu vào duy trì ở mức logic 1. Sau khi tín hiệu điều khiển RESET ở mức tích cực bị loại bỏ thì 82C55A có thể duy trì chế độ hoạt động mà không cần thêm bất kỳ việc khởi tạo nào nữa. Điều này sẽ giúp loại bỏ được các điện trở treo cao hoặc treo thấp trong các thiết kế cho mạch CMOS. Khi kích hoạt chế độ thiết lập thì thanh ghi từ điều khiển sẽ chứa giá trị 9Bh. Trong qúa trình thực hiện chương trình vẫn có thể thay đổi lựa chọn chế độ hoạt động khác nhau, điều này cho phép 82C55 hoạt động một cách đa dạng đáp ứng cho nhiều bài toán ứng dụng khác nhau. Trong qúa trình thanh ghi từ điều khiển đang được viết thì tất cả các cổng được thiết lập hoạt động ở chế độ cổng ra sẽ được khởi tạo bằng zero.

Mode 0 (Vào ra cơ bản): Cấu hình chế độ hoạt động này cung cấp các hoạt động vào ra đơn giản cho cả 3 cổng A, B và C. Dữ liệu được trao đổi trực tiếp và không cần phải có cơ chế bắt tay. Chế độ hoạt động này hỗ trợ các chức năng cụ thể như sau:

- Hai cổng 8-bit và 2 cổng 4-bit
- Bất kỳ cổng nào cũng có thể là cổng vào hoặc cổng ra
- Các đường dữ liệu tín hiệu ra được chốt
- Các đường tín hiệu vào không được chốt
- Có thể cấu hình 16 kiểu hoạt động vào ra khác nhau

Mode 1 (Vào ra có bắt tay): Chế độ hoạt động này cung cấp khả năng truyền dữ liệu tới hoặc đi từ một cổng cụ thể cùng với các tín hiệu bắt tay. Trong chế độ này cổng A, B được sử dụng để truyền dữ liệu và cổng C hoạt động như cổng điều khiển cơ chế động bộ bắt tay. Chế độ hoạt động này cung cấp các chức năng chính sau:

Hai nhóm cổng (Nhóm A và Nhóm B). Mỗi nhóm bao gồm 1 cổng 8-bit và một cổng dữ liệu điều khiển 4-bit.

Cổng dữ liệu 8-bit có thể hoạt động như hoặc là cổng vào hoặc là cổng ra và cả hai chiều dữ liệu đều được chốt.

The 4-bit port is used for control and status of the 8-bit port.

Mode 2 (Bus vào ra hai chiều có bắt tay): Chế độ hoạt động này cung cấp khả năng truyền thông với các ngoại vi hoặc các bus dữ liệu 8-bit cho việc truyền nhận dữ liệu. Các tín hiệu bắt tay được cung cấp để duy trì dòng tín hiệu bus tương tự như chế độ 1. Các cơ chế tạo ngắt cũng có thể được thực hiện ở chế độ này. Một số các chức năng chính hỗ trợ trong chế độ này bao gồm:

- Chỉ sử dụng nhóm A
- Một cổng bus 2 chiều 8-bit (cổng A) và một cổng điều khiển 5-bit (Cổng C)
- Cả hai chiều dữ liệu vào và ra đều đươc chốt.
- Cổng điều khiển 5-bit (Cổng C) được sử dụng cho mục đích điều khiển và trạng thái cho cổng A để trao đổi dữ liệu 2 chiều 8 bit.

2.5 Bus

2.5.1 Bus địa chỉ

Bus địa chỉ là các đường dẫn tín hiệu logic một chiều để truyền địa chỉ tham chiếu tới các khu vực bộ nhớ và chỉ ra dữ liệu được lưu giữ ở đâu trong không gian bộ nhớ. Trong qúa trình hoạt động CPU sẽ điều khiển bus địa chỉ để truyền dữ liệu giữa các khu vực bộ nhớ và CPU. Các địa chỉ thông thường tham chiếu tới các khu vực bộ nhớ

hoặc các khu vực vào ra, hoặc ngoại vi Dữ liệu được lưu ở các khu vực đó thường là 8 bit (1 byte), 16bit, hoặc 32bit tùy thuộc vào cấu trúc từng loại vi xử lý/vi điều khiển.

Hầu hết các vi điều khiển thường đánh địa chỉ dữ liệu theo khối 8bit. Các Các loại vi xử lý 8bit, 16bit và 32bit nói chung cũng đều có thể làm việc trao đổi với kiểu dữ liệu 8bit và 16bit

Chúng ta vẫn thường được biết tới khái niệm địa chỉ truy nhập trực tiếp, đó là khả năng CPU có thể tham chiếu và truy nhập tới trong một chu kỳ bus. Nếu vi xử lý có N bit địa chỉ tức là nó có thể đánh địa chỉ được 2N khu vực mà CPU có thể tham chiếu trực tiếp tới. Qui ước các khu vực được đánh địa chỉ bắt đầu từ địa chỉ 0 và tăng dần đến 2N1.

Hiện nay các vi xử lý và vi điều khiển nói chung chủ yếu vẫn sử dụng phổ biến các bus dữ liệu có độ rộng là 16, 20, 24, hoặc 32bit. Nếu đánh địa chỉ theo byte thì một vi xử lý 16bit có thể đánh địa chỉ được 216 khu vực bộ nhớ tức là 65,536 byte = 64Kbyte. Tuy nhiên có một số khu vực bộ nhớ mà CPU không thể truy nhập trực tiếp tới tức là phải sử dụng nhiều nhịp bus để truy nhập, thông thường phải kết hợp với việc điều khiển phần mềm. Kỹ thuật này chủ yếu được sử dụng để mở rộng bộ nhớ và thường được biết tới với khái niệm đánh địa chỉ trang nhớ khi nhu cầu đánh địa chỉ khu vực nhớ vượt quá phạm vi có thể đánh địa chỉ truy nhập trực tiếp

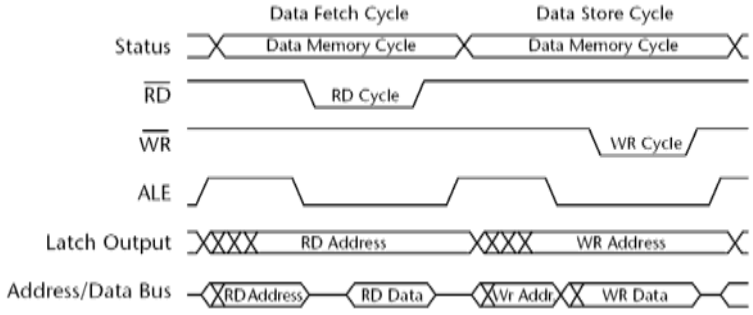
Ví dụ: CPU 80286 có 24bit địa chỉ sẽ cho phép đánh địa chỉ trực tiếp cho 224 byte (16 Mbyte) nhớ. CPU 80386 và các loại vi xử lý mạnh hơn có không gian địa chỉ 32bit sẽ có thể đánh được tới 232 byte (4Gbyte) địa chỉ trực tiếp.

2.5.2 Bus dữ liệu

Bus dữ liệu là các kênh truyền tải thông tin theo hai chiều giữa CPU và bộ nhớ hoặc các thiết bị ngoại vi vào ra. Bus dữ liệu được điều khiển bởi CPU để đọc hoặc viết các dữ liệu hoặc mã lệnh thực thi trong qúa trình hoạt động của CPU. Độ rộng của bus dữ liệu nói chung sẽ xác định được lượng dữ liệu có thể truyền và trao đổi trên bus. Tốc độ truyền hay trao đổi dữ liệu thường được tính theo đơn vị là [byte/s]. Số lượng đường bit dữ liệu sẽ cho phép xác định được số lượng bit có thể lưu trữ trong mỗi khu vực tham chiếu trực tiếp. Nếu một bus dữ liệu có khả năng thực hiện một lần truyền trong 1μs, thì bus dữ liệu 8bit sẽ có băng thông là 1Mbyte/s, bus 16bit sẽ có băng thông là 2Mbyte/s và bus 32bit sẽ có băng thông là 4Mbyte/s. Trong trường hợp bus dữ liệu 8bit với chu kỳ bus là T=1μs (tức là sẽ truyền được 1byte/1chu kỳ) thì sẽ truyền được 1 Mbyte trong 1s hay 2Mbyte trong 2s.

2.5.3 Bus điều khiển

Bus điều khiển phục vụ truyền tải các thông tin dữ liệu để điều khiển hoạt động của hệ thống. Thông thường các dữ liệu điều khiển bao gồm các tín hiệu chu kỳ để đồng bộ các nhịp chuyển động và hoạt động của hệ thống. Bus điều khiển thường được điều khiển bởi CPU để đồng bộ hóa nhịp hoạt động và dữ liệu trao đổi trên các bus. Trong trường hợp vi xử lý sử dụng dồn kênh bus dữ liệu và bus địa chỉ tức là một phần hoặc toàn bộ bus dữ liệu sẽ được sử dụng chung chia sẻ với bus địa chỉ thì cần một tín hiệu điều khiển để phân nhịp truy nhập cho phép chốt lưu trữ thông tin địa chỉ mỗi khi bắt đầu một chu kỳ truyền. Một ví dụ về các chu kỳ bus và sự đồng bộ của chúng trong hoạt động của hệ thống bus địa chỉ và dữ liệu dồn kênh được chỉ ra trong hình sau. Đây là hoạt động điển hình trong họ vi điều khiển 8051 và nhiều loại tương tự.



Hình 2.23 Chu kỳ hoạt động của bus dồn kênh

Câu hỏi cuối chương

1. Nêu các thành phần phần cứng thường có trong một hệ thống nhúng
2. Bộ vi xử lý có thể được tổ chức theo những kiến trúc nào, đặc điểm của mỗi loại kiến trúc đó?
3. Nêu cấu trúc, nguyên lý hoạt động và các loại bộ nhớ Ram
4. Nêu cấu trúc, nguyên lý hoạt động và các loại bộ nhớ Rom
5. Trình bày phương pháp quản lý bộ nhớ trong hệ thống nhúng
6. Nêu sự khác nhau giữa vào ra nối tiếp và vào ra song song
7. Phân tích đặc điểm, vị trí và chức năng của hệ thống BUS

Chương 3 – PHẦN MỀM NHÚNG

3.1 Tổng quan

Phần mềm nhúng là gì ? Phần mềm nhúng là một chương trình được viết, biên dịch trên máy tính và nạp vào một hệ thống khác (gọi tắt là KIT) bao gồm một hoặc nhiều bộ vi xử lý đã được cài sẵn một hệ điều hành, bộ nhớ ghi chép được, các cổng giao tiếp với các phần cứng khác… Mục đích của phần mềm nhúng là nhằm hỗ trợ cho các sản phẩm phần cứng các chức năng hoàn hảo nhất, phục vụ tốt nhất các nhu cầu của người dùng với sự bảo mật về sản phẩm tốt nhất. Phần mềm nhúng có các tính chất sau:

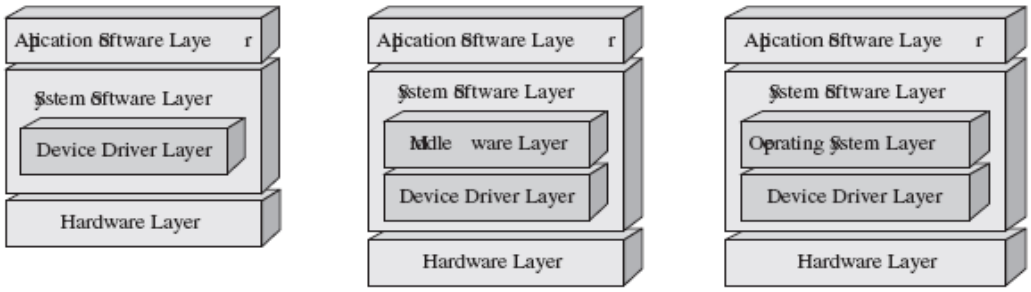
- Phụ thuộc vào hệ điều hành cài sẵn trên KIT
- Phụ thuộc vào các tính năng đặt trưng của từng sản phẩm phần cứng có trong KIT
- Phụ thuộc vào đặc tính của hệ thống

Phần mềm nhúng là sự tích hợp của ngành tin học (phần mềm) với ngành điện tử (phần cứng). Với các các thiết bị điện tử, phần mềm nhúng mang lại nhiều sự hữu ích cần thiết cho người sử dụng và đồng thời giảm chi phí giá thành về phần cứng cho thiết bị. Thế giới ngày này nhắm tới sự tích hợp của ngành tin học với các ngành ứng dụng khác. Sự tích hợp này được thực hiện qua các thiết bị thông minh và phần mềm nhúng là bộ não của các thiết bị đó. Trong thời gian không xa, chúng ta sẽ bước tới kỷ nguyên của "Hậu-PC" (thời đại của hậu máy tính cá nhân) và khi đó thì phần mềm nhúng sẽ là phần đa số của ngành công nghiệp phần mềm. Một số ví dụ phần mềm nhúng: Sản phẩm phần mềm nhúng rất đa dạng, phong phú, thuộc nhiều chủng loại. Có thể lấy các sản phẩm sau làm ví dụ: máy ảnh kỹ thuật số, lò vi ba, máy photocopy, máy in laser, máy FAX, các bảng quảng cáo sử dụng hệ thống đèn LED, màn hình tinh thể lỏng, máy giặt , máy điều hoà nhiệt độ…

3.2 Trình điều khiển thiết bị

3.2.1 Tổng quan

Hầu hết các thành phần phần cứng đều cần một vài loại phần mềm để quản lý và điều khiển. Các phần mềm giao tiếp trực tiếp và điều khiển các thành phần phần cứng này được gọi là các chương trình điều khiển thiết bị. Tất cả những hệ thống nhúng đều có các trình điều khiển thiết bị tại lớp phần mềm hệ thống của mình. Trình điều khiển thiết bị là một thư viện phần mềm có nhiệm vụ khởi động phần cứng, quản lý việc truy nhập tới phần cứng cả các lớp phần mềm cao hơn. Trình điều khiển thiết bị có vị trí trung gia giữa phần cứng và hệ điều hành, các chương trình ứng dụng.



Hình 3.1 Mô hình hệ hống nhúng và phần mềm điều khiển thiết bị

Trình điều khiển thiết bị được chia thành hai loại là architecture-specific hoặc generic. Một trình điều khiển là architecture-specific nếu nó quản lý các phần cứng tích hợp trong bảng mạch chính và liên kết với bộ xử lý chính. Các nhiệm của trình điều khiển thiết bị này bao gồm việc khởi tạo và điều khiển các thành phần trong bộ xử lý chính như bộ nhớ chính, đơn vị điều khiển bộ nhớ, các phần cứng liên quan đến tính toán các phép toán dấu phẩy động. Một trình điều khiển là generic nếu nó quản lý các phần cứng không nằm trong bảng

mạch chính, không liên kết với bộ xử lý chính. Trong trình điều khiển generic thường có một phần chương trình của trình điều khiển architecture-specific bởi vì bộ xử lý chính là đơn vị điều khiển trung tâm cần thiết phải truy nhập đến mọi thành phần phần cứng, trình điều khiển generic có thể được sửa đổi và cấu hình để thực thi trên nhiều nền tảng phần cứng khác nhau có sự tương đồng về phần cứng. Các trình điều khiển generic bao gồm mã lệnh khởi động và quản lý việc truy nhập đến các thành phần còn lại của bảng mạch như các bus (I2C, PCI, PIMCIA...), các bộ nhớ ngoài chip (level-2+ cache, Flash ...) và các chip ngoài (Ethernet, Rs 232, Display, Mouse...)

3.2.2 Ngắt

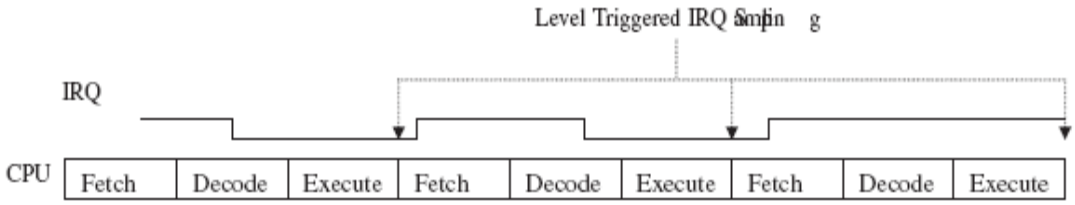
Ngắt là các tín hiệu được tạo ra bởi một số sự kiện trong quá trình thực thi các chỉ thị lệnh của bộ xử lý. Các ngắt có thể là không đồng bộ, với các thiết bị phần cứng ngoài như ngắt reset, power... hoặc có thể là đồng bộ với những hoạt động liên quan đến chỉ thị lệnh như các lời gọi hàm hoặc các chỉ thị khác. Những tín hiệu này khiến bộ xử lý dừng hoạt động hiện tại của dòng chỉ lệnh và khởi động thủ tục xử lý ngắt.

Phần mềm xử lý ngắt trong bộ xử lý và phần cứng quản lý các ngắt thường bao gồm các trình điều khiển cho việc xử lý ngắt. Các chức nang mà các trình điều khiển cung cấp có thể là:

- Interrupt Handling Startup: khởi động các ngắt cứng (ví dụ như bộ điều khiển ngắt, kích hoạt các ngắt ...) khi hệ thống khởi động hoặc reset.
- Interrupt Handling Shutdown: cấu hình các ngắt (điều khiển ngắt, ngừng kích hoạt các ngắt ...) về trạng thái power off.
- Interrupt Handling Disable: cho phép các phần mềm khác có thể tác động vào các ngắt (disable). Tuy nhiên không cho phép với các ngắt Non-Maskable.
- Interrupt Handling Enable: cho phép các phần mềm khác có thể tác động vào các ngắt (enable).
- Interrupt Hadler Servicing: điều khiển mã lệnh của bản thân ngắt mà được thực thi ngay sau khi luồng chỉ thị lệnh chính dừng.

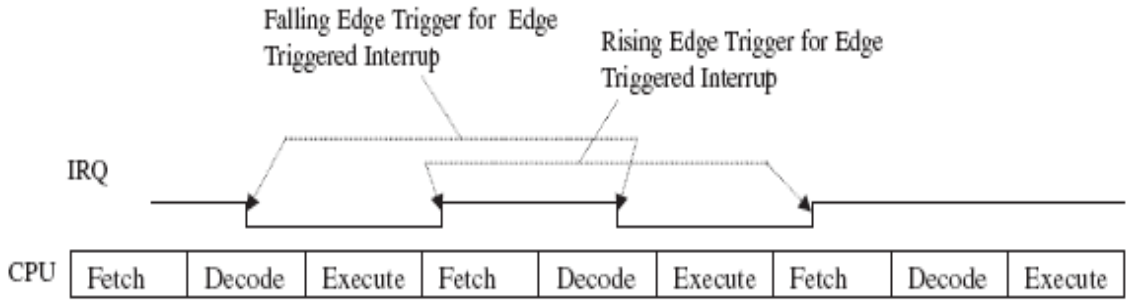
Có ba loại ngắt chính đó là software, internal hardware và external hardware. Các ngắt Software được sinh ra bởi các chỉ thị lệnh bên trong luồng chỉ thị đang thực thi bởi bộ xử lý. Các ngắt Internal hardware được khởi tạo bởi các sự kiện là kết quả của các vấn đề (thường là lỗi) với luồng chỉ thị đang được thực hiện bởi các đặc trưng của phần cứng như lỗi tràn bộ nhớ, lỗi chia cho không, lỗi sai chỉ lệnh .... Các ngắt sinh ra bởi những sự kiện nội tại này thường được gọi là các ngoại lệ exception. Cuối cùng, các ngắt external hardware được tạo ra bởi các thành phần phần cứng như hệ thống bus, các thiết bị IO...

Với các ngắt được sinh ra bởi các nguyên nhân bên ngoài, bộ xử lý sẽ kết nối qua chân IRQ (Interrupt Request Level) với bộ xử lý ngắt hoặc trực tiếp với các thành phần trên mạch phục vụ cho ngắt này. Ngắt có thể được phát sinh theo hai cách: level-triggered và edge-triggered. Ta nói ngắt phát sinh theo level-triggered khi tín hiệu IRQ của nó có mức độ nhất định (HIGH hoặc LOW). Những ngắt thế này được xử lý khi CPU tìm ra một yêu cầu xử lý trên đường IRQ sau mỗi lần thực hiện một chỉ thị lệnh nào đó.



Hình 3.2 Ngắt Level-triggered

Các ngắt edge-triggered được sinh ra khi có một sự thay đổi trên đường tín hiệu IRQ (từ LOW sang HIGH hoặc ngược lại). Một khi được sinh ra, những ngắt như vậy sẽ được đưa vào CPU chờ xử lý.



Hình 3.3 Ngắt Edge-triggered

Tại thời điểm một IRQ của bộ xử lý nhận được tín hiệu chỉ thị có một ngắt, ngắt đó sẽ được xử lý bởi các cơ chế xử lý ngắt bên trong hệ thống. Các cơ chế này bao gồm cả phần mềm và phần cứng. Dưới khía cạnh phần cứng, bộ điều khiển ngắt có thể được tích hợp vào mạch hoặc bộ xử lý để có thể trao đổi trực tiếp với các thành phần phần mềm liên quan. Với các hệ thống không có bộ điều khiển ngắt, các đường yêu cầu ngắt được kết nối trực tiếp với bộ xử lý chính và các thủ tục ngắt được điều khiển bằng phần mềm với các mạch điện tử.

3.2.3 Bộ nhớ

Trong khi thực tế cấu tạo vật lý bộ nhớ là một mảng hai chiều (ma trận), mỗi một phần tử nhớ có một địa chỉ hàng và cột duy nhất thì bộ xử lý lại coi bộ nhớ như là một mảng một chiều và được tham chiếu như là một Memory Map (MM). Trong MM, mỗi phần tử nhớ là một hàng các bytes, và số lượng các bytes trên mỗi hàng phụ thuộc vào độ rộng của bus dữ liệu (8 bit, 16 bit, 32 bit, 64 bit...). Điều này nói theo một cách khác là phụ thuộc vào các thanh ghi của kiến trúc xử lý. Khi bộ nhớ vật lý được tham chiếu từ phần mềm, nó được coi là bộ nhớ logic và có phần tử nhớ là byte. Bộ nhớ logic được hình thành từ tất cả các bộ nhớ vật lý (thanh ghi, ROM, RAM) trong toàn bộ hệ thống nhúng.

Address Range	Accessed Device	Port Width
0x00000000 - 0x003FFFFF	Flash PROM Bank 1	32
0x00400000 - 0x007FFFFF	Flash PROM Bank 2	32
0x04000000 - 0x043FFFFF	DRAM 4 Mbyte (1Meg × 32-bit)	32
0x09000000 - 0x09003FFF	MPC Internal Memory Map	32
0x09100000 - 0x09100003	BCSR - Board Control & Status Register	32
0x10000000 - 0x17FFFFFF	PCMCIA Channel	16

Hình 3.4 Ví dụ về Memory Management

Hệ thống phần mềm phải đảm bảo cung cấp cho bộ xử lý trong hệ thống khả năng truy nhập vào tất cả các phần của bản đồ nhớ MM. Các phần mềm phải quản lý được các bộ nhớ trong bộ xử lý và trên mạch cũng như là quản lý các cơ chế bộ nhớ phần cứng như các chương trình điều khiển của các hệ thống con. Các hệ thống nhớ con bao gồm tất cả các thành phần quản lý bộ nhớ như Memory Controller và MMU cũng như các kiểu bộ nhớ có trong MM. Các chức năng mà trình điều khiển bộ nhớ phải cung cấp bao gồm:

- Memory Subsystem Startup: Phần khởi tạo bộ nhớ khi hệ thống được cấp nguồn hoặt reset.
- Memory Subsystem Shutdown: Cấu hình lại thông tin để chuyển hệ thống nhớ vào trạng thái Power-off.
- Memory Subsystem Disable: Cho phép các phần mềm khác có thể vô hiệu hóa một hệ thống nhớ nào đó (ví dụ như cache)
- Memory Subsystem Enable: Cho phép các phần mềm khác có thể kích hoạt hệ thống nhớ nào đó.
- Memory Subsystem Write: Ghi dữ liệu vào một vùng nhớ xác định.
- Memory Subsystem Read: Đọc dữ liệu từ một vùng nhớ xác định.

Bất kể sự khác nhau giữa các kiểu dữ liệu, mọi dữ liệu ta có thể coi là một dãy các byes. Trong khi một lần truy cập bộ nhớ bị giới hạn về kích thước của bus dữ liệu thì ta có các cơ chế để truy cập bộ nhớ với từng khối block, gọi là segments. Với việc truy nhập này ta phải cung cấp các thông số liên quan đến segment number và offset để xác định địa chỉ vật lý của khối nhớ.

Trật tự mà trong đó các bytes được lấy ra hoặt ghi vào phụ thuộc vào kiến trúc byte ordering của hệ thống. Hai kiểu kiến trúc thông dụng đó là little endian và big endian. Với little endian, các byte được đọc và ghi theo thứ tự byte thấp trước, ngược lại với big endian.

3.2.4 Bus

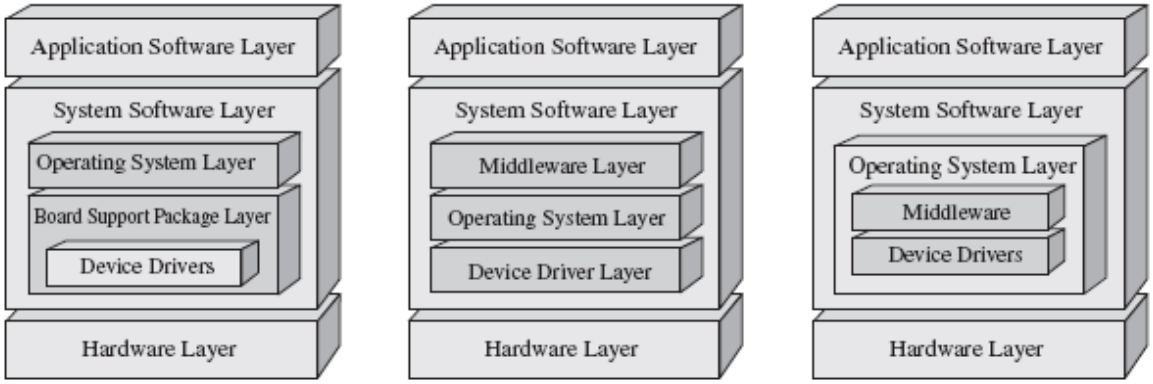
Như ta đã biết, các thông tin liên quan đến bất kì một hệ thống BUS nào bao gồm: protocol định nghĩa cách thức mà các thiết bị truy nhập vào bus, các qui định để các thiết bị kết nối với nhau thông qua bus và các tín hiệu trên bản thân bus. Các giao thức Bus được cung cấp với trình điều khiển bus với các chức năng chính sau đây:

- Bus Startup: khởi động các thông số ban đầu khi hệ thống được cấp nguồn hoạt động hoặc reset
- Bus Shutdown: cấu hình bus để chuyển về trạng thái Power Off
- Bus Disable: cho phép các phần mềm khác vô hiệu hóa Bus ngay lập tức
- Bus Enable: cho phép các phần mềm khác kích hoạt Bus ngay lập tưc
- Bus Acquire: cho phép các phần mềm khác có quyền sử dụng Bus
- Bus Release: cho phép các phần mềm khác giải phóng Bus
- Bus Read: cho phép các phần mềm khác đọc dữ liệu từ Bus
- Bus Write: cho phép các phần mềm khác ghi dữ liệu lên Bus
- Bus Install: cho phép các phần mềm khác được cài đặt các thiết bị để mở rộng Bus
- Bus Uninstall: cho phép các phần mềm khác được loại bỏ các thiết bị Bus

3.3 Hệ điều hành trong các hệ thống nhúng

3.3.1 Tổng quan

Với hệ thống nhúng, hệ điều hành OS chỉ là một lựa chọn, nghĩa là ta có thể gặp những hệ thống nhúng có hoặc không có hệ điều hành. Các hệ điều hành có thể được sử dụng trên bất kỳ một bộ xử lý nào. Như trong hình sau, hệ điều hành được đặt trên phần cứng, các trình điều khiển thiết bị hoặc BSP (Board Support Package).



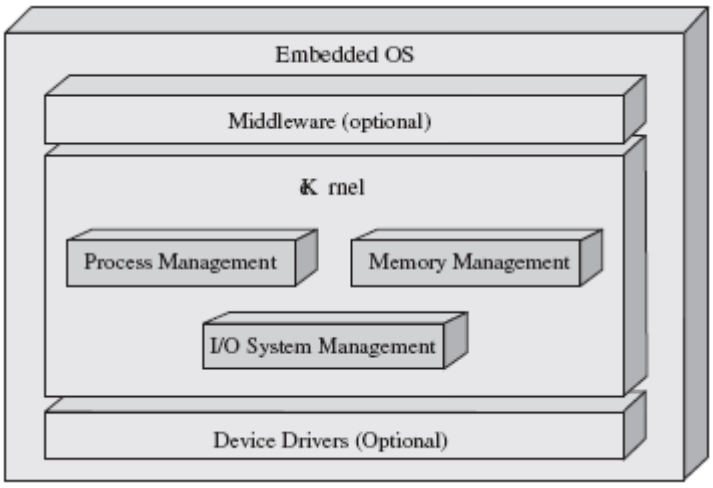
Hình 3.5 Mô hình hệ thống nhúng và hệ điều hành

Hệ điều hành là tập hợp các thư viện phần mềm phục vụ cho hai mục đích chính: cung cấp sự trong suốt cho các phần mềm hoạt động trên hệ điều hành đó, làm cho công việc phát triển các ứng dụng được dễ dàng hơn và quản lý các tài nguyên phần cứng và phần mềm để đảm bảo cho toàn hệ thống hoạt động một cách hiệu quả, tin cậy. Trong khi các hệ điều hành nhúng khác nhau ở các thành phần xử lý, hầu hết các hệ điều hành này đều có một nhân Kernel rất hạn chế. Kernel là thành phần chứa các chức năng cơ bản nhất của hệ điều hành, bao gồm:

Quản lý tiến trình: cung cấp các cơ chế quản lý các phần mềm trong hệ thống nhúng. Một chức năng nhỏ thường có trong quản lý tiến trình đó là việc quản lý các ngắt và ngoại lệ. Tất cả các ngắt, ngoại lệ, lỗi sinh ra bởi các tiến trình đều phải được quản lý một cách hiệu quả sao cho chúng được xử lý chính xác và các tiến trình sinh ra những ngoại lệ này được theo dõi đúng mức.

Quản lý bộ nhớ: Không gian nhớ trong hệ thống nhúng được chia sẻ bởi tất cả các tiến trình, vì vậy việc truy nhập và một phần tử nhớ phải được quản lý chặt chẽ. Một chức năng chính trong quản lý bộ nhớ là việc quản lý an ninh hệ thống, chức năng này cho phép một phần của hệ thống có thể cô lập để tránh ảnh hưởng tới sự hoạt động của toàn bộ hệ thống.

Quản lý vào ra: Các thiết bị vào ra được chia sẻ bởi nhiều tiến trình và do đó cũng phải được quản lý như là bộ nhớ. Một trong những chức năng chính của quản lý vào ra là quản lý việc truy nhập các file giữa các tiến trình.

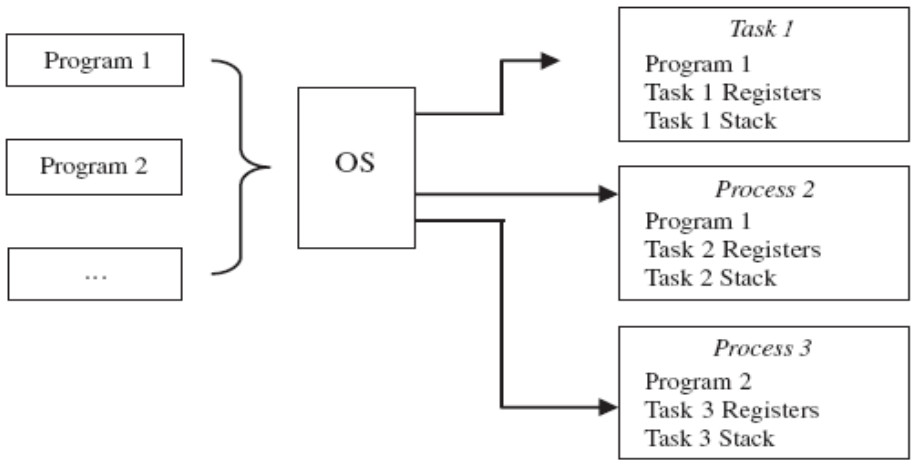


Hình 3.6 Mô hình hệ thống nhúng

3.3.2 Tiến trình

Để hiểu rõ cách thức hệ điều hành quản lý các tài nguyên phần cứng và phần mềm ta phải hiểu rõ cách hệ điều hành nhìn nhận về hệ thống. Hệ điều hành phân biệt giữa một chương trình và sự thực thi của chương trình đó. Chương trình đơn giản là một dãy các chỉ thị lệnh tĩnh và thụ động. Việc thực thi thực tế của chương trình là động, trong đó có rất nhiều sự thay đổi theo thời gian. Một tiến trình (task) được tạo bởi hệ điều hành trong đó có chứa tất cả những thông tin liên quan đến sự thực thi của chương trình (ví dụ stack, PC, mã nguồn, dữ liệu ...). Điều này nghĩa là chương trình chỉ là một phần của tiến trình.

Các hệ điều hành nhúng quản lý tất cả các phần mềm bằng cách sử dụng các tiến trình, ta có thể có đơn tiến trình hoặc đa tiến trình. Trong môi trường đơn tiến trình, chỉ có một tiến trình được thực thi trong một thời điểm. Ngược lại, có thể có nhiều tiến trình cùng thực thi trong môi trường đa tiến trình. Ta thấy rõ ràng là đơn tiến trình sẽ không yêu cầu các cơ chế thuật toán phức tạp như đa tiến trình. Với đa tiến trình ta phải đảm bảo các tiến trình độc lập với nhau và không ảnh hưởng đến nhau ngoại trừ trường hợp được xác định rõ trong mã lệnh. Mô hình đa tiến trình cung cấp cơ chế để chia nhỏ chương trình thành các thành phần mà có thể được xử lý song song.



Hình 3.7 Mô hình đa tiến trình

3.3.3 Quản lý tiến trình

Lý thuyết lập lịch

Định nghĩa: Lý thuyết Scheduling đề cập tới vấn đề lập lịch những tiến trình có thể thực hiện đồng thời với các yếu tố thời gian deadline và quyền ưu tiên.

Tài nguyên CPU là hằng số, nếu hệ thống có nhiều tiến trình ta phải trả lời câu hỏi là liệu có thể phân chia thời gian, lập lịch cho các tiến trình đó sao cho chúng hoàn thành trong đúng thời gian đã định. Nếu chúng ta thực hiện lần lượt từng tiến trình, từng tiến trình thì không có khái niệm về lập lịch Schedule bởi vì một tiến trình chỉ thực hiện khi tiến trình kia đã kết thúc, Schedule chỉ đề cập với những tiến trình có thể thực hiện song song với nhau. Lý thuyết về Schedule là rất rộng và đã phát triển từ những năm 60, trong chương trình này ta chỉ đề cập đến một phần rất nhỏ của nó. Ta sẽ lập lịch cho các tiến trình dựa trên hai yếu tố đó là khoảng thời gian deadline của từng tiến trình, độ ưu tiên của chúng và phương pháp lập lịch mà ta sẽ vận dụng là phương pháp "Rate monotonic Scheduling".

Phương pháp Rate Monotonic

Rate Monotonic một phương pháp rất phổ biến được phát triển bởi Liu và Layland vào năm 1973 tại ACM, phương pháp này đã được cài đặt trong ngôn ngữ Ada 9x.

Trước hết ta định nghĩa về các dãy đơn điệu, một dãy đơn điệu là một dãy các số được sắp xếp theo thứ tự tăng hoặc giảm. Ví dụ dãy

2, 5, 12, 27 ,99

là dãy đơn điệu (tăng) còn dãy số sau không phải là dãy đơn điệu

3, 5, 8, 6, 10, 9

Định nghĩa

Phương pháp Rate Monotonic Scheduling là phương pháp trong đó quyền ưu tiên cao hơn sẽ được gán cho những tiến trình, tiến trình mà thời gian hoàn thành dealine là nhỏ hơn. Trước khi tiếp tục nghiên cứu về RMS, chúng ta hãy tìm hiểu một số giả thiết sau.

Cho T_i là khoảng thời gian duy trì của tiến trình trong hệ thống (bằng một số nguyên lần chu kỳ hệ thống)

Cho C_i là khoảng thời gian tiến trình yêu cầu CPU

D_i là khoảng thời gian deadline của tiến trình. Chúng ta coi $D_i=T_i$ (nghĩa là tiến trình hoàn thành trong khoảng thời gian duy trì của nó)

Chúng ta định nghĩa U_i (Utilization Ratio - Tỉ lệ sử dụng) như sau:

$$U_i = \frac{C_i}{T_i}$$

U_i định nghĩa tỉ lệ của thời gian thực hiện và thời gian duy trì của tiến trình, ta có thể thấy hiển nhiên giá trị có thể chấp nhận của U_i là 1.

Các điều kiện cần của RMS

Trong phần này ta sẽ tìm hiểu các điều kiện cần để một tập hợp các tiến trình có khả năng lập lịch. Chú ý rằng những điều kiện này không phải là điều kiện đủ.

Điều kiện 1 - Đơn sử dụng

$$\forall i : T_i : C_i \le T_i$$

Đây là điều kiện hiển nhiên, nếu thời gian duy trì tiến trình nhỏ hơn thời gian hoàn thành tiến trình đó thì ta không thể lập lịch dù trong hệ thống chỉ có duy nhất tiến trình đó. Trong trường hợp này ta phải thay đổi thuật toán hoặc nâng cấp hiệu năng của CPU để giảm thời gian hoàn thành C_i.

Điều kiện 2 - Đa sử dụng

$$\sum_{i=1}^{n} (C_i / T_i) \le 1$$

Điều kiện này chỉ ra rằng tổng của các tỉ lệ sử dụng là không vượt quá 1. Điều kiện 1 chỉ ra rằng thời gian để một tiến trình sử dụng là không vượt quá 1, bởi vì nếu $\sum U_i = 1$ nghĩa là CPU đã được sử dụng 100% thời gian, do vậy nó không thể thực hiện thêm một tiến trình nào khác. Do vậy tổng tỉ lệ sử dụng của tất cả các tiến trình phải nhỏ hơn hoặc bằng 1.

Bây giờ ta sẽ quan tâm đến một khái niệm quan trọng - deadline - trước khi phân tích điều kiện thứ 3. Deadline được định nghĩa bởi 2 thời điểm, thời điểm thứ nhất là khi bắt đầu tiến trình và thứ hai là thời điểm mà tiến trình đó cần hoàn thành. Ta coi deadline của một tiến trình bằng với thời gian duy trì của nó, nghĩa là tiến trình có thể hoàn thành công việc của mình trước khi thời gian duy trì kết thúc.

Xem xét một tiến trình j với chu kỳ Tj, tiến trình này cần thời gian là Cj để kết thúc. Chu kỳ của j có thể không đồng thời với thời điểm bắt đầu chu kỳ trừ trường hợp nó là tiến trình có độ ưu tiên cao nhất. So đó sẽ có một pha trễ P trước khi một tiến trình có thể được thực hiện.

Để tiến trình j có thể được hoàn thành, pha trễ tối đa là Tj - Cj, nếu pha trễ lớn hơn giá trị này thì tiến trình không thể hoàn thành trước chu kỳ tiếp theo.

Tương tự, nếu pha trễ P nhỏ hơn thời gian Tj - Cj nhưng nếu các tiến trình có độ ưu tiên cao hơn chiếm thời gian lớn hơn Tj - Cj thì tiến trình j cũng không thể hoàn thành trước chu kỳ tiếp theo.

Giả sử cho 2 tiến trình i, j, Ti > Tj, do đó độ ưu tiên của j là cao hơn i. Tiến trình j sẽ giành quyền ưu tiên của tiến trình i nhiều nhất là Ti/Tj lần. Như trong ví dụ trước, tiến trình 1 sẽ giành quyền ưu tiên của tiến trình 3 số lần là 200/50.

Do đó, thời gian mà tiến trình j lấy của tiến trình j sẽ là

$$\frac{T_i}{T_j} * C_j$$

bởi vì tiến trình j cần Cj thời gian để hoàn thành.

Tổng quát, với một tiến trình i, tất cả các tiến trình có mức ưu tiên cao hơn sẽ giành thời gian của tiến trình i là:

$$\sum_{j=1}^{i-1} (T_i / T_j) * C_j \le (T_i - C_j)$$

Để hiểu rõ hơn về công thức trên, ta phân tích về biểu đồ ở phần trước. Ta thấy là tiến trình 3 (period = 200) bị tiến trình 1 giành quyền ưu tiên tất cả là 4 lần. Do đó thời gian mà T_1 đã giành là:

[200/50] * 10 = 40

Tương tự như vậy, thời gian T_2 đã giành của T_3 là

[200/100] * 20 = 40

Ta thấy là trong 200 đơn vị thời gian dành cho T_3, tổng thời gian mà T_1 và T_2 đã giành là 40 + 40 = 80, thời gian còn lại là 200 - 80 = 120.

Thời gian cần thiết để T_3 hoàn thành là 50 đơn vị, thời gian thực tế giành cho t3 là 120 đơn vị, do đó ta thấy tập hợp T_1, T_2, T_3 là có khả năng lập lịch.

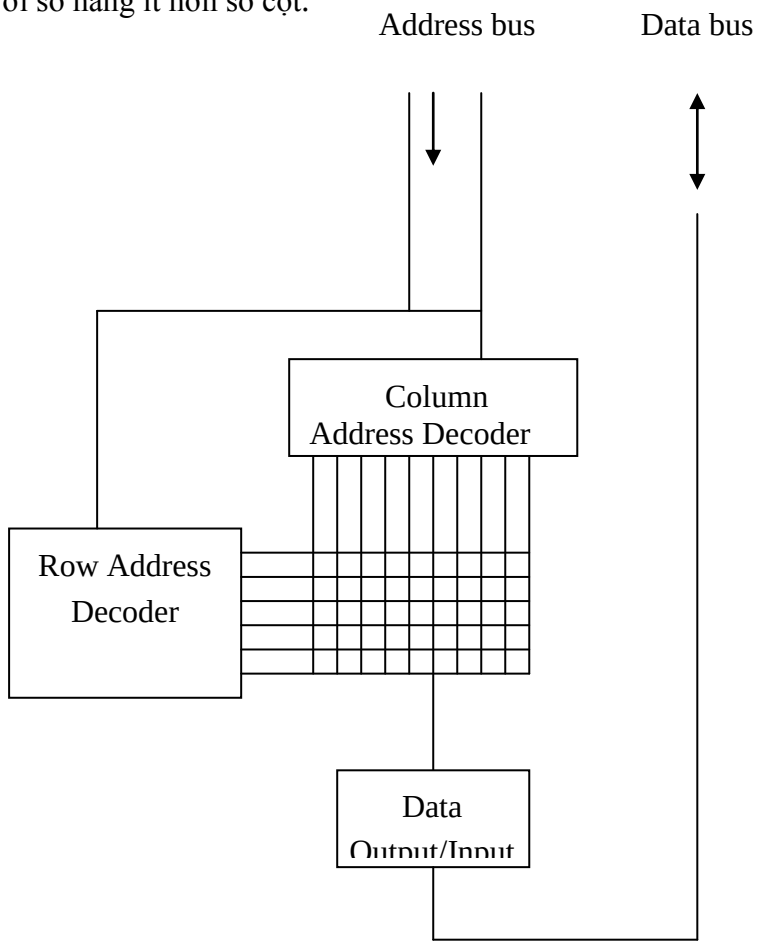
3.3.4 Quản lý bộ nhớ

Truy cập bộ là thủ tục thường dùng để đọc và ghi vào bộ nhớ. Thủ tục này thường được sử dụng để khiểm soát mỗi truy cập vào bộ nhớ bao gồm bộ điều khiển bộ nhớ để tạo ra các tín hiệu chính xác để xác định vị trí nhớ cần được truy cập. Điều này được thực hiện đựa trên việc truy cập dữ liệu từ chương trình. Dữ liệu này hiển thị qua các đường dẫn dữ liệu được kết nối với bộ xử lý hoặc bất cứ thiết bị khác mà được yêu cầu.

Các chip nhớ được sắp xếp dữ liệu theo hàng hoặc cột, Cho Vd như, 16 MB chip có thể được truy cập như một khối 4M x 4. Điều này có ý nghĩa là có 4M (4,194,304) địa chỉ với mỗi 4 bit ; vì vậy chúng có các 4,194,304 các vùng nhớ khác nhau _đôi khi được gọi là các ô nhớ _ Mỗi một ô chưa 4 bit dữ liệu. 4,194,304 là ngang bằng với 2^22, nó có nghĩa như 22 bit được yêu cầu tới các địa chỉ được đặt cho các vị trí nhớ. Như vậy, trong lý thuyết 22 dòng địa chỉ được yêu cầu.

Tuy nhiên, trong thực thế, các chip nhớ không có nhiều các dòng địa chỉ. Chúng được thay thế sắp xếp một cách logic như một "hình vuông" của các hàng, các cột _ đôi khi được gọi một cách tương ứng là bitlines và wordline.11 bit cấp thấp được xem xét như "hàng ", và 11 bit cấp cao như " cột". Địa chỉ hàng đầu được đưa tới chip, và sau đó là địa chỉ cột. Vd, hãy giả sử rằng chúng ta muốn truy cập tới vùng nhớ 3,780,514 trong chip này. Điều này tương ứng với một địa chỉ nhị phân của "11100110101111110100010". Thứ nhất, " 11110100010" sẽ đưa ra lựa chọn " hàng" và sau đó " 11100110101" sẽ đưa ra lựa chọn cột. Sự kết hợp này lựa chọn đúng một vùng địa chỉ nhớ 3,780,514. Hoạt động của hàng và cột sau đó đưa dữ liệu tới chúng qua các đường dẫn bằng dữ liệu trung gian.

Hình 3.1 biểu diễn ở một mức khái niệm, một ví dụ về truy cập bộ nhớ bằng lưới tọa độ ở hàng 8*8 và cột. Chú ý rằng lưới tọa độ không vuông, và trong thực tế nó thường là hình chữ nhật với số hàng ít hơn số cột.

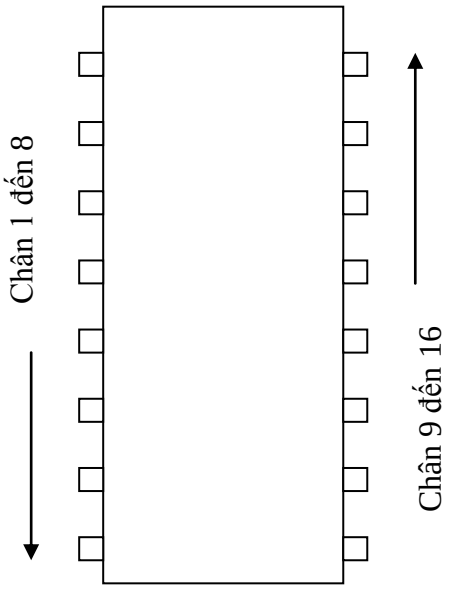


Hình 3.8 Truy nhập bộ nhớ

Việc này tương tự như một ô nhớ riêng biệt trên một bảng tính được lựa chọn và thiết lập : # 34 hàng, và sau đó tìm tới cột " J" và tìm ô nhớ " J34". Tương tự như vậy, Vd làm thế nào để những người sành cờ tướng tạo ra các bước cờ bởi Vishwanathan Anand ở NewYork ngày hôm qua ? Một cách cơ bản, chỉ cần đánh dấu các hàng từ 1 đến 8 trong bàn cờ và các cột từ a tới g. Hầu hết các chuyển dịch có thể được biểu diễn bởi một loạt những sự kết hợp chữ số trong bảng chữ cái.

Bây giờ chúng ta hãy cùng trở lại thế giới của các chip. Nếu chúng ta áp dụng ý nghĩa của lý thuyết này, chúng ta có thể thiết kế các bộ nhớ chip, theo phương thức này cả hai đều phức tạp và chậm hơn việc chỉ cài đặt một chân cắm chỉ trên mỗi chip và yêu cầu các địa chỉ là đơn trị ? Tại sao không đặt 22 chân cắm địa chỉ trên con chip ? Câu trả lời có thể không là ngạc nhiên tới nhiều người : đó là về chi phí cuối cùng. Một cách đặc biệt là khi nhiều các hệ thống không có những ràng buộc phức tạp về thời gian thực, chúng vẫn có thể hoạt động với sự trì hoãn của việc truy cập bộ nhớ nếu chúng làm cho hệ thống đơn giản và rẻ hơn, Bằng việc sử dụng các phương thức hàng / cột, Nó có thể làm giảm một lượng lớn các chân cắm ở trên các DRAM chip. Ở đây, 11 địa chỉ các chân cắm được yêu cầu thay thế cho 22, Tuy nhiên, chúng ta nên chú ý để bố trí xung tín hiệu được yêu cầu đến các chip nhớ và việc truy cập thiết bị luôn được chronised cái mà chúng ta đang mong chờ. Chân cắm tín hiệu này luôn được gọi để thăm dò hoặc lựa chọn chip. Một điều chắc chắn rằng : những thứ không đổi, phải được đưa tới địa chỉ trong hai " khối ". nhưng bằng cách giữ các con chip nhỏ hơn và đầu nhập ít hơn sẽ cho ta lợi ích về điện năng tiêu thụ và không gian (bởi vì giảm số lượng các

chân cắm). Giảm tiêu thụ điện năng hơn nữa dẫn tới tăng tốc độ của chip, giảm bớt đi một phần tốc độ truy cập. Hình 3.2 biểu diễn chip nhớ điển hình với 8 dòng địa chỉ và 2 dòng dữ liệu.



Hình 3.9 Chip nhớ

Các chức năng của các chân cắm là:

Vcc: Đây là các chân cắm through để chip để được nạp điện từ nguồn cho chức năng.

DND: Đây là các chân cắm Ground phải được yêu cầu cho bất cứ mạch điện nào.

WE: Cho phép ghi. Khi chân cắm này được bảo vệ, điều này có nghĩa rằng các chip được lựa chọn cho việc ghi. Điều đó cũng có nghĩa rằng dữ liệu sẽ được gửi tới thông qua chân cắm Din và định rõ cho vùng dữ liệu này ở các chân cắm từ A0 tới A7. Trong trường hợp này, vùng dữ liệu lại được ghi lại thông qua địa chỉ các chân cắm từ A0 tới A7. Dữ liệu dưa ra thông qua chân cắm Dout.

RAS và CAS: Như chúng ta đã ghi chú ở trước đó, các hàng và các cột bên trong vùng nhớ được đặt một địa chỉ trên một thời gian. Do đó, trong thời điểm này cần đưa ra một cách thức ra lệnh cho chip, những dòng địa chỉ xác định một số hàng hoặc một số cột. Khi RAS được thiết lập cao, địa chỉ một hàng được xác định, otherwise địa chỉ của một cột được tính toán. Như chúng ta đã đề cập tới trước đó, các chân cắm thêm vào yêu cầu phải được đồng bộ hóa để giảm số chân cắm cho địa chỉ đường dẫn.

Với quá trình ghi

- Địa chỉ của các ô nhớ để ghi được đặt lên địa chỉ các chân cắm thông qua các địa chỉ đường dẫn . Điều này được thực hiện trong lần thiết lập đầu tiên của RAS và sử dụng số hàng tương ứng theo việc thiết lập CAS và sử dụng số cột trên các địa chỉ đường dẫn.
- Set the Write Enable pin : Bit mà cần được lưu trữ trong con chip để gửi Data In pin thông qua các đường dẫn dữ liệu.
- Việc lựa chọn con chip được tác động để lựa chon bộ nhớ chip : Khi hầu hết các hoạt động được thực hiên đồng thời, một bit trên chân cắm Din được viết bên trong con chip ở địa chỉ xác định bởi đường dẫn địa chỉ

Với quá trình đọc

- Địa chỉ của bit để đọc được đặt trên đia chỉ của các chân cắm thông qua các địa chỉ của bus.Các chân cắm RAS và CAS thường được sử dụng một cách tương thích.

- Việc lựa chọn con chip được turned high để lựa chọn bộ nhớ chip.
- Cấp phép ghi lên chân cắm được turned chậm đến mưc chip biết được nó đang đọc từ đâu.
- Khi hầu hết các tác động được thể hiện cùng một lúc, dữ liệu xuất hiện trên chân cắm Dout từ các địa chỉ xác định bởi địa chỉ đường dẫn.

Trong thực tế, bộ nhớ được truy cập ít nhất ở một byte tại một thời gian, và không có bit nào cũng trong thời gian đó. Điều này được thực hiện bằng cách xếp chồng mỗi con chip tới các khối và kết hợp các phân lớp bit dữ liệu từ 8 chip. Khi các chip cần được gán địa chỉ, các,việc lựa chọn chip được cho phép và các địa chỉ tương đương được xác định trên hầu hết các dòng địa chỉ. Tùy thuộc vào khả năng của các đường dẫn dữ liệu, mỗi khối lại một lần nữa được xếp chồng lên nhau để tạo ra hàng triệu các khối mà có thể cung cấp dữ liệu trong bội số của byte.

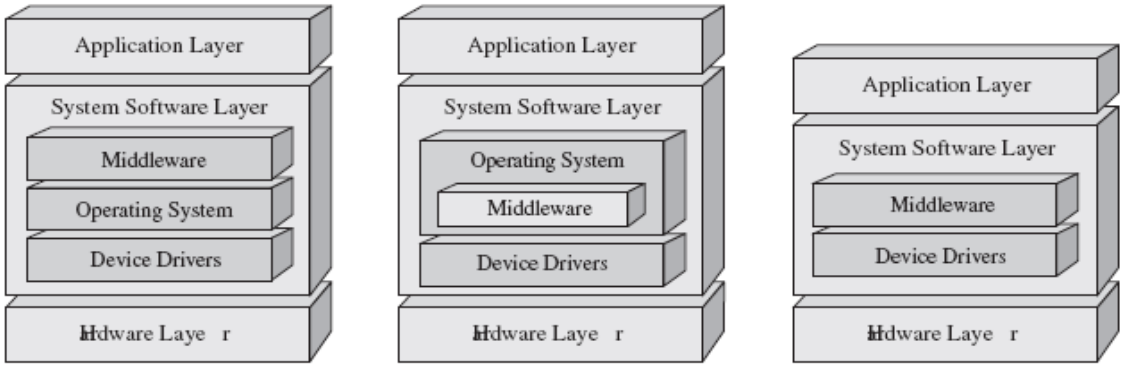
Thời gian truy nhập được định nghĩa chuẩn xác trong nanogiây (ns). Tính khả dụng của bộ nhớ này có thời gian truy nhập khác nhau từ 5 đến 70 nanogiay(ns).

3.4 Phần mềm ứng dụng

Trong phần này ta tìm hiểu về hai loại phần mềm là middleware và application. Sự khác nhau giữa hai loại này là không thực sự rõ ràng, middleware là loại phần mềm được tách ra khỏi lớp ứng dụng do một số lý do, một nguyên nhân là phần mềm này được tích hợp như là một phần của gói hệ điều hành. Nguyên nhân khác là do yêu cầu sử dụng lại của các hệ thống ứng dụng khác làm giảm thời gian và chi phí phát triển.

3.4.1 Middleware

Trong hầu hết các trường hợp, middleware là bất kỳ hệ thống phần mềm nào mà không phải là hệ điều hành, trình điều khiển thiết bị hoặc phần mềm ứng dụng. Tóm lại, trong một hệ thống nhúng, middleware là hệ thống phần mềm tồn tại trên trình điều khiển thiết bị hoặc trên hệ điều hành và có thể được tích hợp vào hệ điều hành.



Hình 3.10 Middleware trong hệ thống nhúng

Middleware thường là phần mềm trung gian giữa phần mềm ứng dụng và kernel hoặc device driver. Middleware cũng là phần mềm cung cấp các hàm dịch vụ cho các ứng dụng phần mềm khác. Cụ thể hơn, middleware là một lớp phần mềm được sử dụng bởi các thiết bị nhúng với các ứng dụng phần mềm với mục đích tạo ra sự linh hoạt, tính an ninh, liên kết giữa các ứng dụng. Một trong những mục đích chính sử dụng middleware là cho phép giảm độ phức tạp của ứng dụng. Middleware tác động đến mọi lớp của hệ thống nhúng .

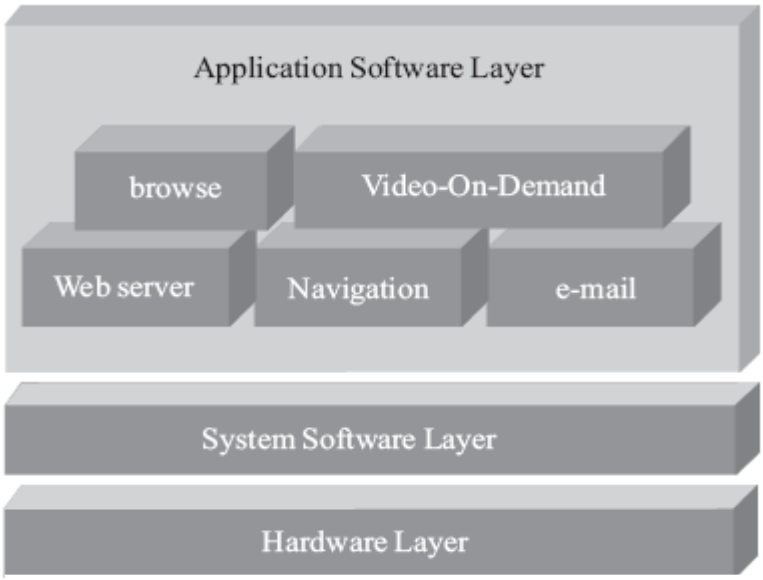
Có rất nhiều loại middleware, ví dụ như các loại middleware hướng thông điệp (message oriented middleware MOM), triệu gọi thủ tục từ xa (remote procedure calls RPCs), kết nối cơ sở dữ liệu, các giao thức mạng.... Tuy nhiên, hầu hết các loại middleware đều rơi vào một trong hai dạng sau:

- General Purpose: chúng được cài đặt trong nhiều loại thiết bị khác nhau, ví dụ như các giao thức mạng bên trên tầng device driver và bên dưới tầng ứng dụng trong mô hình OSI, các hệ thống file hoặc các dạng máy ảo như JVM.
- Market specific: chúng được phát triển cho một họ hệ thống nhúng nào đó, ví dụ như các hệ thống chuẩn Tivi ...

Ngoài ra ta còn có thể chia ra thành các loại middleware đóng hay mở tùy theo các loại giấy phép bản quyền đi kèm với nó.

3.4.2 Application

Một dạng phần mềm thông dụng ta gặp trong các hệ thống nhúng là application. Như trong hình dưới, application có vị trí ở phần đỉnh của các lớp phần mềm, nó phụ thuộc, quản lý và thực thi bởi các phần mềm hệ thống. Đó là các phần mềm trong lớp ứng dụng đặc trưng cho loại thiết bị của hệ thống nhúng bởi vì các chức năng của ứng dụng biểu diễn mục đích chính của hệ thống và thực hiện hầu hết các tương tác với người sử dụng.



Hình 3.11 Application trong hệ thống nhúng

Các ứng dụng nhung co the được phan loại theo general purpose hoặc market specific như ta đã đề cập ở trên.

Câu hỏi cuối chương

1. Sự khác nhau giữa middleware và application
2. Nêu cách phân loại middleware và application
3. Phân tích các điều kiện cần của RMS
4. Phân tích phương pháp Rate Monotonic
5. Trình bày về hệ điều hành trong hệ thống nhúng
6. Trình bày về trình điều khiển thiết bị
7. Trình bày về trình điều khiển ngắt
8. Trình bày về trình điều khiển bộ nhớ
9. Trình bày về trình điều khiển bus

Chương 4 – THIẾT KẾ HỆ THỐNG NHÚNG THEO CÁC HỌ VI XỬ LÝ

4.1 Tổng quan

Sự ra đời và phát triển nhanh chóng của kỹ thuật vi điện tử mà đặc trưng là kỹ thuật vi xử lí đã tạo ra một bước ngoặt quan trọng trong sự phát triển của khoa học tính toán, điều khiển và xử lí thông tin. Kỹ thuật vi xử lí đóng một vai trò rất quan trọng trong tất cả các lĩnh vực của cuộc sống và khoa học kỹ thuật, đặc biệt là lĩnh vực Tin học và Tự động hóa.

Năm 1971, hãng Intel đã cho ra đời bộ vi xử lí (microprocessor) đầu tiên trên thế giới tên gọi là Intel-4004/4bit , nhằm đáp ứng nhu cầu cấp thiết của một công ty kinh doanh là hãng truyền thông BUSICOM. Intel-4004 là kết quả của một ý tưởng quan trọng trong kỹ thuật vi xử lí số. Đó là một kết cấu logic mà có thể thay đổi được chức năng của nó bằng chương trình ngoài chứ không phát triển theo hướng tạo ra một cấu trúc cứng chỉ thực hiện một số chức năng nhất định như trước đây.

Sau đó, các bộ vi xử lí mới liên tục được đưa ra thị trường và ngày càng được phát triển, hoàn thiện hơn trong các thế hệ sau :

Vào năm 1972, hãng Intel đưa ra bộ vi xử lí 8-bit đầu tiên với tên Intel-8008/8bit. Từ 1974 đến 1975 , Intel chế tạo các bộ vi xử lí 8-bit 8080 v 8085A. Cũng vào khoảng thời gian này, một loạt các hãng khác trên thế giới cũng đã cho ra đời các bộ vi xử lí tương tự như : 6800 của Motorola với 5000 tranzitor, Signetics 6520, 1801 của RCA, kế đến là 6502 của hãng MOS Technology và Z80 của hãng Zilog.

Năm 1978 xuất hiện Intel 8086 là loại bộ xi xử lý 16 bit với 29.000 tranzitor, Motorola 68000 tích hợp 70.000 tranzitor, APX 432 chứa 120.000 tranzitor. Bộ vi xử lý của Hewlet Pakard có khoảng 450.000 tranzitor. Từ năm 1974 đến 1984 số tranzitor tích hợp trong một chip tăng khoảng 100 lần.

Năm 1983, Intel đưa ra bộ vi xử lý 80286 dung trong các máy vi tinh họ AT (Advanced Technology). 80286 sử dụng I/O 16 bit, 24 đường địa chỉ và không gian nhớ địa chỉ thực 16MB. Năm 1987, Intel đưa ra bộ vi xử lý 80386 32-bit. Năm 1989 xuất hiện xuất hiện bộ vi xử lý Intel 80486 là cải tiến của Intel 80386 với bộ nhớ ẩn và mạch tính phép toán đại số dấu phẩy động. Năm 1992, xuất hiện Intel 80586 còn gọi là Pentium 64 bit chứa 4 triệu tranzitor.

Độ phức tạp, sự gọn nhẹ về kích thước và khả năng của các bộ vi điều khiển được tăng thêm một bậc quan trọng vào năm 1980 khi Intel công bố chip 8051, bộ vi điều khiển đầu tiên của họ vi điều khiển MCS-51. So với 8048, chip 8051 chứa trên 60.000 transistor bao gồm 4K byte ROM, 128 byte RAM, 32 đường xuất nhập, 1 port nối tiếp và 2 bộ định thời 16-bit – một số lượng mạch đáng chú ý trong một IC đơn.

Từ các bộ vi xử lý ban đầu chỉ là các bộ xử lý trung tâm trong một hệ thống, không thể hoạt động nếu thiếu các bộ phận như RAM, ROM, bo mạch chủ... các hãng đã phát triển các bộ vi xử lý này lên thành các bộ vi điều khiển để phục vụ các mục đích riêng biệt, khác nhau trong công nghiệp. Một bộ vi điều khiển là một hệ vi xử lí thật sự được tổ chức trong một chip (trong một vỏ IC) bao gồm một bộ vi xử lí (microprocessor), bộ nhớ chương trình (ROM), bộ nhớ dữ liệu (RAM), tuy không bằng dung lượng RAM ở các máy vi tính nhưng đây không phải là một hạn chế vì các bộ vi điều khiển được thiết kế cho một mục đích hoàn toàn khác, ngoài ra trên chip còn có bộ xử lý số học-logic (ALU) cùng với các thanh ghi chức năng, các cổng vào/ra, cơ chế điều khiển ngắt, truyền tin nối tiếp, các bộ định thời... Hiện nay, các bộ vi điều khiển được sử dụng rất rộng rãi và ngày càng được chuẩn hóa để có thể sử dụng rộng rãi trong các ngành công nghiệp, có mặt trong nhiều máy móc, trong các hàng tiêu dùng.

Các công việc được thực hiện bởi các bộ vi điều khiển thì không mới. Điều mới là các thiết kế hiện thực với ít thành phần hơn so với các thiết kế trước đó. Các thiết kế trước đó đòi hỏi phải vài chục hoặc vài trăm IC để hiện thực nay chỉ cần một ít thành phần trong đó bao

gồm bộ vi điều khiển. Số thành phần được giảm bớt, hiệu quả trực tiếp của tính khả lập trình của các bộ vi điều khiển và độ tích hợp cao trong công nghệ chế tạo vi mạch, thường chuyển thành thời gian phát triển ngắn hơn, giá thành khi sản xuất thấp hơn, công suất tiêu thụ thấp hơn và độ tin cậy cao hơn. Vấn đề ở đây là tốc độ. Các giải pháp dựa trên bộ vi điều khiển không bao giờ nhanh bằng giải pháp dựa trên các thành phần rời rạc. Những tình huống đòi hỏi phải đáp ứng thật nhanh (cỡ nsec) đối với các sự kiện (thường chiếm thiểu số trong các ứng dụng) sẽ được quản lý tồi khi dựa vào các bộ vi điều khiển.

Tuy nhiên trong vài ứng dụng, đặc biệt là các ứng dụng liên quan đến con người, các khoảng thời gian trễ tính bằng nsec, µsec hoặc thậm chí msec là không quan trọng. Việc giảm bớt các thành phần là một điều lợi như đã đề cập, các thao tác trong chương trình điều khiển làm cho thiết kế có thể thay đổi bằng cách thay đổi phần mềm. Điều này có ảnh hưởng tối thiểu đến chu kỳ sản xuất. Do đó các bộ vi điều khiển có thể được ứng dụng rộng rãi trong các ứng dụng phục vụ con người.

Để có thể hiểu rõ hơn về các bộ vi điều khiển, chúng ta sẽ tìm hiểu về một số các họ vi điều khiển của một số hãng điện tử điển hình đang được sử dụng rộng rãi trong khoa học kỹ thuật và đời sống.

4.2 Họ vi xử lý AT89C

4.2.1 Tổng quan

AT9C51 là một hệ vi tính 8 bit đơn chíp CMOS có hiệu suất cao,công suất nguồn tiêu thụ thấp và có 4Kbyte bộ nhớ ROM Flash xoá được lập trình được. Chip này được sản xuất dựa vào công nghệ bộ nhớ không mất nội dung có độ tích hợp cao của Atmel.

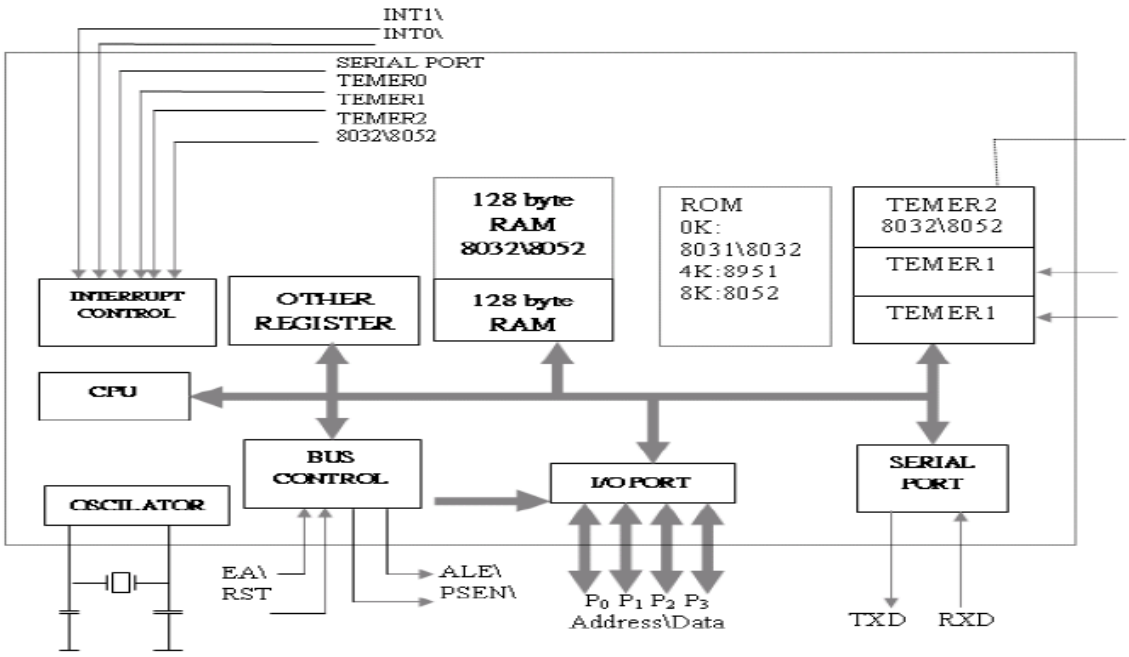
Chip AT89C51 cũng tương thích với tập lệnh và các chân ra của chuẩn công nghiệp MCS-51. Flash trên chip này cho phép bộ nhớ chương trình được lập trình lại trên hệ thống hoặc bằng bộ lập trình bộ nhớ không mất nội dung qui ước. Bằng cách kết hợp một CPU linh hoạt 8 bit với Flash trên một chip đơn thể, Atmel 89C51 là một hệ vi tính 8 bit đơn chip mạnh cho ta một giải pháp có hiệu quả về chi phí và rất linh hoạt đối với các ứng dụng điều khiển. AT89C51 có các đặc trưng sau: 4Kbyte Flash, 128 byte Ram, 32 đường xuất nhập, hai bộ định thời / đếm 16 bit, một cấu trúc ngắt 2 mức ưu tiên và 5 nguyên nhân ngắt, một port nối tiếp song công, mạch dao động và tạo xung clock trên chip.

Ngoài ra AT8951 được thiết kế với logic tĩnh cho hoạt đông có tần số giảm xuống 0 và hỗ trợ hai chế độ tiết kiệm năng lượng được lựa chọn bằng phần mềm. Chế độ nghĩ dừng CPU trong khi vẫn cho phép RAM, các bộ định thời /đếm, port nối tiếp và hệ thống ngắt tiếp tục hoạt động. Chế độ nguồn giảm duy trì nội dung của RAM nhưng không cho mạch dao động cung cấp xung clock nhằm vô hiệu hoá các hoạt động khác của chip cho đến khi có reset cứng tiếp theo.

Các đặc điểm của 8951 được tóm tắt như sau:

- 4 KB bộ nhớ có thể lập trình lại nhanh, có khả năng tới 1000 chu kỳ ghi xoá
- Tần số hoạt động từ: 0Hz đến 24 MHz
- 2 bộ Timer/counter 16 Bit
- 128 Byte RAM nội.
- Port xuất /nhập I/O 8 bit.
- Giao tiếp nối tiếp.
- 64 KB vùng nhớ mã ngoài
- 64 KB vùng nhớ dữ liệu ngoại.
- Xử lý Boolean (hoạt động trên bit đơn).
- 210 vị trí nhớ có thể định vị bit.
- 4ms cho hoạt động nhân hoặc chia.

4.2.2 Kiến trúc họ vi xử lý AVR



Hình 4.1 Sơ đồ khối AT89C

Mô tả các chân

- VCC: chân cung cấp điện.
- GND: chân nối đất.
- Port 0: gồm 8 chân 32-39 (P0.0…P0,7): Port 0 là port có 2 chức năng. Trong các thiết kế cỡ nhỏ không dùng bộ nhớ mở rộng nó có chức năng như các đường IO. Đối với các thiết kế cỡ lớn có bộ nhớ mở rộng, nó được kết hợp giữa bus địa chỉ và bus dữ liệu.
- Port1: chân 1-8 (P1.0…P1.7): Port 1 là port IO. Có thể dùng cho giao tiếp với các thiết bị ngoài nếu cần. Port 1 không có chức năng khác, vì vậy chúng chỉ được dùng cho giao tiếp với các thiết bị bên ngoài.
- Port 2: chân 21-28 (P2.0….P2.7): Port 2 là port có tác dụng kép. Được dùng như các đường xuất nhập hoặc byte cao của bus địa chỉ đối với các thiết bị dùng bộ nhớ mở rộng.
- Port 3: chân 10-17 (P3.0…P3.7): Port 3 là port xuất nhập 8 bit 2 chiều có các điện trở kéo lên bên trong. Các chân của port này có nhiều chức năng, các công dụng chuyển đổi có lien hệ với các đặc tính đặc biệt của 8951
- RST: Ngõ vào reset. Mức cao trên chân này trong 2 chu kỳ máy trong khi bộ dao động đang hoạt động sẽ reset AT89C51.
- ALE/ PROG: Xung của ngõ ra cho phép chốt địa chỉ ALE (address latch enable) cho phép chốt byte thấp của địa chỉ trong thời gian truy xuất bộ nhớ ngoài. Chân này cũng được dùng làm ngõ vào xung lập trình (PROG) trong thời gian lập trình Flash. Khi hoạt động bình thường, xung của ngõ ra ALE luôn luôn có tần số bằng 1/6 tần số của mạch dao động trên chip, có thể được sử dụng cho các mục đích định thời từ bên ngoài và tạo xung clock. Tuy nhiên cần lưu ý là một xung ALE sẽ bị bỏ qua trong mỗi chu kỳ truy xuất bộ nhớ dữ liệu ngoài. Khi cần, hoạt động cho phép chốt byte thấp của địa chỉ sẽ được vô hiệu hoá bằng cách set bit 0 của thanh ghi chức năng đặc biệt có địa chỉ byte là 8EH. Khi bit này được set, ALE chỉ tích cực tring thời gian thực thi lệnh MOVX hoặc MOVC. Ngược lại chân này sẽ được kéo lên mức cao. Việc set bit không cho phép hoạt động chốt byte thấp của địa chỉ sẽ không có tác dụng nếu bộ vi điều khiển đang chế độ thực thi chương trình ngoài.

- PSEN : Chân cho phép bộ nhớ chương trình PSEN (program store enable) điều khiển truy xuất bộ nhớ chương trình ngoài. Khi AT89C51 đang thực thi chương trình trong bộ nhớ chương trình ngoài, PSEN tích cực 2 lần cho mỗi chu kỳ máy,ngoại trừ trường hợp 2 tác động của PSEN bị bỏ qua cho mỗi truy xuất bộ nhớ dữ liệu ngoài.
- EA / Vpp: Chân cho phép truy xuất bộ nhớ ngoài EA (external access enable) phải được nối với GND để cho phép chip vi điều khiển tìm nạp lệnh từ các vị trí nhớ của bộ nhớ chương trình ngoài, bắt đầu từ địa chỉ 0000H cho đến FFFFH. Tuy nhiên cần lưu ý là nếu có bit khoá 1(clock bit 1) được lập trình, EA sẽ được chốt bên trong khi reset. EA nên nối với Vcc để thực thi chương trình trong chip. Chân EA / Vpp còn nhận điện áp cho phép lập trình Vpp trong thời gian lập trình cho Flash, điện áp này cấp cho các bộ phận có yêu cầu điện áp 12V.
- XTAL 1: Ngõ vào đến mạch khuyếch đại đảo dao động và ngõ vào đến mạch tạo xung clock bên trong chip.
- XTAL 2: Ngõ ra từ mạch khuyết đại đảo của mạch dao động.

4.2.3 Tập lệnh

AT89C bao gồm các lệnh sau:

- Các lệnh số học.
- Lệnh logic.
- Dịch chuyển dữ liệu.
- Luân lý.
- Rẽ nhánh chương trình.

Từng kiểu lệnh được mô tả như dưới đây:

Các lệnh số học (Arithmetic Instrustion):

ADD A, <src, byte>

SUBB A, <src, byte>

INC <byte>

DEC <byte>

MULL AB : (A) ← LOW [(A) x (B)] ; có ảnh hưởng cờ OV

(B) ← HIGH [(A) x (B)] ; cờ Cary được xóa.

DIV AB : (A) ← Integer Result of [(A)/(B)]; cờ OV

(B) ← Remainder of [(A)/(B)]; cờ Carry xóa

Các hoạt động logic (Logic Operation) :

ANL <dest - byte> <src - byte>

ORL <dest - byte> <src - byte>

XRL <dest - byte> <src - byte>

CLR A : (A) ← 0

CLR C : (C) ← 0

CLR Bit : (Bit) ← 0

RL A : Quay vòng thanh ghi A qua trái 1 bit

RLC A : Quay vòng thanh ghi A qua trái 1 bit có cờ Carry

RR A : Quay vòng thanh ghi A qua phải 1 bit

RRC A : Quay vòng thanh ghi A qua phải 1 bit có cờ Carry

SWAP A : Đổi chổ 4 bit thấp v 4 bit cao của A cho nhau (A3 ← A0) ← (A7 ← A4).

Các lệnh rẽ nhánh :

JC rel : Nhảy đến "rel" nếu cờ Carry C = 1.

JNC rel : Nhảy đến "rel" nếu cờ Carry C = 0.

JB bit, rel : Nhảy đến "rel" nếu (bit) = 1.

JNB bit, rel : Nhảy đến "rel" nếu (bit) = 0.

JBC bit, rel : Nhảy đến "rel" nếu bit = 1 v xĩa bit.

ACALL addr11 : Lệnh gọi tuyệt đối trong page 2K.

LCALL addr16 : Lệnh gọi di chương trình con trong 64K.

RET : Kết thúc chương trình con trở về chương trình chính.

RETI : Kết thúc thủ tục phục vụ ngắt quay về chương trình chính hoạt động tương tự như RET.

AJMP Addr11 : Nhảy tuyệt đối không điều kiện trong 2K.

LJMP Addr16 : Nhảy đi không điều kiện trong 64K. Hoạt động tương tự lệnh LCALL.

SJMP rel : Nhảy ngắn không điều kiện trong (-128 127) byte

JMP @ A + DPTR:Nhảy không điều kiện đến địa chỉ (A) + (DPTR)

(PC) ← (A) + (DPTR)

JZ rel : Nhảy đến A = 0. Thực hành lệnh kế nếu A 0.

JNZ rel : Nhảy đến A 0. Thực hành lệnh kế nếu A = 0.

CJNE A, direct, rel : So sánh và nhảy đến A ← direct

(A) < (direct) ← C = 1

(A) > (direct) ← C = 0

(A) = (direct). Thực hành lệnh kế tiếp

CJNE A, # data, rel : Tương tự lệnh CJNE A, direct, rel.

CJNE Rn, # data, rel : Tương tự lệnh CJNE A, direct, rel.

CJNE @ Ri, # data, rel : Tương tự lệnh CJNE A, direct, rel.

DJNE Rn, rel : Giảm Rn nhảy nếu Rn # 0.

DJNZ direct, rel : Tương tự lệnh DJNZ Rn, rel.

Các lệnh dịch chuyển dữ liệu :

Tất cả các lệnh dịch chuyển đều khôg ảnh hưởng đến cờ. Hoạt động của từng lệnh được tóm tắt như sau :

MOV A,Rn : (A) ← (Rn)

MOV A, direct : (A) ← (direct)

MOV A, @ Ri : (A) ← ((Ri))

MOV A, # data : (A) ← # data

MOVX A, @ Ri : (A) ← ((Ri))

MOVX A, @ DPTR : (A) ← ((DPTR))

MOVX @ Ri, A : ((Ri)) ← (A)

MOVX @ DPTR, A : ((DPTR)) ← (A)

PUSH direct : Cất dữ liệu vào Stack

POP direct : Lấy từ Stack ra direct

XCH A, Rn : Đổi chổ nội dung của A với Rn (A) ← (Rn)

XCH A, direct : (A) ← (direct)

XCH A, @ Ri : (A) ← ((Ri))

XCHD A, @ Ri : Đổi chổ 4 bit thấp của (A) với ((Ri))

Các lệnh luận lý (Boolean Instruction) :

CLR C : Xóa cờ Carry xuống 0. Có ảnh hưởng cờ Carry.

CLR BIT : Xóa bit xuống 0. Không ảnh hưởng cờ Carry

SET C : Set cờ Carry ln 1. Có ảnh hưởng cờ Carry.

SET BIT : Set bit lên 1. Không ảnh hưởng cờ Carry.

CPL C : Đảo bit cờ Carry. Có ảnh hưởng cờ Carry.

CPL BIT : Đảo bit. Không ảnh hưởng cờ Carry.

ANL C, BIT : (C) ← (C) AND (BIT): Có ảnh hưởng cờ Carry.

ANL C, /BIT : (C) ← (C) AND NOT (BIT): không ảnh hưởng cờ Carry

ORL C, BIT : (C) ← (C) OR (BIT): Tác động cờ Carry.

ORL C, /BIT : (C) ← (C) OR NOT (BIT): Tác động cờ Carry.

MOV C, BIT : (C) ← (BIT): Cờ Carry bị tác động.

MOV BIT, C : (BIT) ← (C): Không ảnh hưởng cờ Carry.

4.2.4 Sự thực thi

Trong phần này chúng ta sẽ tìm hiểu cách hoạt động, thực thi của các bộ định thời timer trong vi xử lý này.

AT89C51 có 2 bộ timer:

- Timer 0: là một bộ đếm lên tuần tự 16 bit, giá trị đếm chứa trong 2 thanh ghi TH0, TL0.
- Timer 1: là một bộ đếm tuần tự 16 bit chứa trong TH1 và TL1.

Bit	Tên	Timer	Mô tả
7	GATE	1	Khi GATE = 1, Timer chỉ làm việc khi INT1=1
6	C/T	1	Bit cho đếm sự kiện hay ghi giờ
			C/T = 1 : Đếm sự kiện
			C/T = 0 : Ghi giờ đều đặn
5	M1	1	Bit chọn mode của Timer 1
4	M0	1	Bit chọn mode của Timer 1
3	GATE	0	Bit cổng của Timer 0
2	C/T	0	Bit chọn Counter/Timer của Timer 0
1	M1	0	Bit chọn mode của Timer 0
0	M0	0	Bit chọn mode của Timer 0

Bảng 4.1 Các Timer trong AT89C

Hai bit M0 và M1 của TMOD để chọn mode cho Timer 0 hoặc Timer 1.

M1	M0	Mode	Mô tả
0	0	0	Mode Timer 13 bit (mode 8048)
0	1	1	Mode Timer 16 bit
1	0	2	Mode tự động nạp 8 bit
1	1	3	Mode Timer tách ra : Timer 0 : TL0 là Timer 8 bit được điều khiển bởi các bit của Timer 0. TH0 tương tự nhưng được điều khiển bởi các bit của mode Timer 1. Timer 1 : Được ngừng lại.

Bảng 4.2 Các Timer Mode trong AT89C

Thanh ghi TCON (Timer control)

Thanh ghi điều khiển bao gồm các bit trạng thái và các bit điều khiển bởi Timer 0 và Timer 1. Thanh ghi TCON có bit định vị. Hoạt động của từng bit được tóm tắt như sau:

Bit	Symbol	Bit Address	Mô tả
TCON.7	TF1	8FH	Cờ tràn Timer 1 được set bởi phần cứng ở sự tràn, được xóa bởi phần mềm hoặc bởi phần cứng khi ác vectoxử lý đến thủ tục phục vụ ngắt ISR
TCON.6	TR1	8EH	Bit điều khiển chạy Timer 1 được set hoặc xóa bởi phần mềm để chạy hoặc ngưng chạy Timer.
TCON.5	TF0	8DH	Cờ tràn Timer 0(hoạt động tương tự TF1)
TCON.4	TR0	8CH	Bit điều khiển chạy Timer 0 (giống TR1)
TCON.3	IE1	8BH	Cờ kiểu ngắt 1 ngoài. Khi cạnh xuống xuất hiện trên INT1 thì IE1 được xóa bởi phần mềm hoặc phần cứng khi CPU định hướng đến thủ tục phục vụ ngắt ngoài.
TCON.2	IT1	8AH	Cờ kiểu ngắt 1 ngoài được set hoặc xóa bằng phấn mềm bởi cạnh kích hoạt bởi sự ngắt ngoài.
TCON.1	IE0	89H	Cờ cạnh ngắt 0 ngoài
TCON	IT0	88H	Cờ kiểu ngắt 0 ngoài.

Các chế độ hoạt động của Timer:

- Mode 0: gồm 8 bit của thanh ghi THx, 5 bit của thanh ghi TLx. (Rất ít được sử dụng)
- Mode 1: bộ Timer là bộ đếm gồm 16 bit.
  - ✓ Gồm 8 bit của thanh ghi THx và 8 bit của thanh ghi TLx.
  - ✓ Đếm giá trị từ 0 đến 2^{16}-1=65535
  - ✓ Với mode 1 bộ timer không tự động nạp lại giá trị đếm sau mỗi lần tràn. Do đó sau mỗi lần tràn ta phải nạp lại giá trị đặt cho timer.

- Mode 2: là chế độ tự động nạp lại. Bộ timer là bộ đếm 8 bit chứa trong thanh ghi THx. Thanh ghi THx không thực hiện nhiệm vụ đếm, mà chỉ nạp lại giá trị cho thanh ghi TLx sau mỗi lần tràn.
- Mode 3 là mode Timer tách ra và là sự khác biệt cho mỗi Timer.
  - ✓ Timer 0 ở mode 3 được chia là 2 timer 8 bit. TL0 và TH0 hoạt động như những Timer riêng lẻ với sự tràn sẽ set các bit TL0 và TF1 tương ứng.
  - ✓ Timer 1 bị dừng lại ở mode 3, nhưng có thể được khởi động bởi việc ngắt nó vào một trong các mode khác. Chỉ có nhược điểm là cờ tràn TF1 của Timer 1 không bị ảnh hưởng bởi các sự tràn của Timer 1 bởi vì TF1 được nối với TH0.
  - ✓ Mode 3 cung cấp 1 Timer ngoại 8 bit là Timer thứ ba của 8951. Khi vào Timer 0 ở mode 3, Timer có thể hoạt động hoặc tắt bởi sự ngắt nó ra ngoài và vào trong mode của chính nó hoặc có thể được dùng bởi Port nối tiếp như là một máy phát tốc độ Baud, hoặc nó có thể dùng trong hướng nào đó mà không sử dụng Interrupt

4.2.5 Thiết kế ứng dụng

Trong phần này ta sẽ tìm hiểu một số ứng dụng, bài tập đơn giản sử dụng tập lệnh của AT89C.

Nối mạch thí nghiệm: Nối JP7: P1_CPU trên module Microcontroler với JP27: DATA_LED trên module LED, các led tương ứng từ led1 đến led8 sẽ nối với các bit P1.0 đến P1.7, các led đều tác động ở mức cao.

Chương trình 1: Chớp tắt 8 led vô hạn lần

```
MAIN:
  MOV P1,#0FFH ; P1 <- 11111111B, các led đều sáng
  CALL DELAY ; gọi chương trình trì hoãn DELAY
  MOV P1,#00H ; P1 <- 00000000B, các led đều tắt
  CALL DELAY
  LJMP MAIN ; nhảy đến MAIN để lập lại quá trình vô hạn
  DELAY:
  PUSH 06 ; cất nội dung R6 vào ngăn xếp
  PUSH 07 ; cất nội dung R7 vào ngăn xếp
  MOV R6,#255
  LAP:
  MOV R7,#255
  DJNZ R7,$ ; X: DJNZ R7,X
  DJNZ R6, LAP
  POP 07 ; lấy lại giá trị cũ của R7 trong ngăn xếp
  POP 06 ; lấy lại giá trị cũ của R0 trong ngăn xếp
  RET ; kết thúc chương trình con.
END
```

Chương trình 2: giống nội dung chương trình 01 nhưng lặp lại quá trình 10 lần.

```
MOV Ri, # <SỐ LẦN LẶP (1->255)>
NHÃN:
- Lệnh 1
- Lệnh 2
- Lệnh N
```

```
  DJNZ Ri, <NHÃN>
```
- Chương trình:
```
  MOV R7,#10
MAIN:
  MOV P1,#0FFH ; P1 <- 11111111B, các led đều sáng
  CALL DELAY ; gọi chương trình trì hoãn DELAY
  MOV P1,#00H ; P1 <- 00000000B, các led đều tắt
  CALL DELAY
  DJNZ R7,MAIN ;
  SJMP $ ; “dừng chương trình”
  DELAY:
  PUSH 06 ; cất nội dung R6 vào ngăn xếp
  PUSH 07 ; cất nội dung R7 vào ngăn xếp
  MOV R6,#255
  LAP:
  MOV R7,#255
  DJNZ R7,$ ; X: DJNZ R7,X
  DJNZ R6, LAP
  DJNZ R6, LAP
  POP 07 ; lấy lại giá trị cũ của R7 trong ngăn xếp
  POP 06 ; lấy lại giá trị cũ của R0 trong ngăn xếp
  RET ; kết thúc chương trình con.
END
```
Chương trình 3: viết chương trình dịch một led sáng từ D1 - D8:
```
MAIN:
  MOV A,#01H
BEGIN:
  MOV P1, A
  RL A
  CALL DELAY
  LJMP BEGIN
  DELAY:
  PUSH 06 ; cất nội dung R6 vào ngăn xếp
  PUSH 07 ; cất nội dung R7 vào ngăn xếp
  MOV R6,#255
  LAP:
  MOV R7,#255
  DJNZ R7,$ ; X: DJNZ R7,X
  DJNZ R6, LAP
  POP 07 ; lấy lại giá trị cũ của R7 trong ngăn xếp
  POP 06 ; lấy lại giá trị cũ của R0 trong ngăn xếp
  RET ; kết thúc chương trình con.
END
```

Chương trình 4: viết chương trình dịch một led sáng từ D8 - D1:

```
MAIN:
  MOV A,#80H
BEGIN:
  MOV P1, A
  RR A
  CALL DELAY
  LJMP BEGIN
  DELAY:
  PUSH 06 ; cất nội dung R6 vào ngăn xếp
  PUSH 07 ; cất nội dung R7 vào ngăn xếp
  MOV R6,#255
  LAP:
  MOV R7,#255
  DJNZ R7,$ ; X: DJNZ R7,X
  DJNZ R6, LAP
  POP 07 ; lấy lại giá trị cũ của R7 trong ngăn xếp
  POP 06 ; lấy lại giá trị cũ của R0 trong ngăn xếp
  RET ; kết thúc chương trình con.
END
```

Chương trình 5: viết chương trình sáng dần các led từ D1 - D8:

```
MAIN:
  MOV A,#01H
BEGIN:
  SETB C
  MOV P1, A
  RLC A
  CALL DELAY
  JNC BEGIN
  LJMP MAIN
  DELAY:
  PUSH 06 ; cất nội dung R6 vào ngăn xếp
  PUSH 07 ; cất nội dung R7 vào ngăn xếp
  MOV R6,#255
  LAP:
  MOV R7,#255
  DJNZ R7,$ ; X: DJNZ R7,X
  DJNZ R6, LAP
  POP 07 ; lấy lại giá trị cũ của R7 trong ngăn xếp
  POP 06 ; lấy lại giá trị cũ của R0 trong ngăn xếp
  RET ; kết thúc chương trình con.
END
```

4.3 Họ vi xử lý AVR

4.3.1 Tổng quan

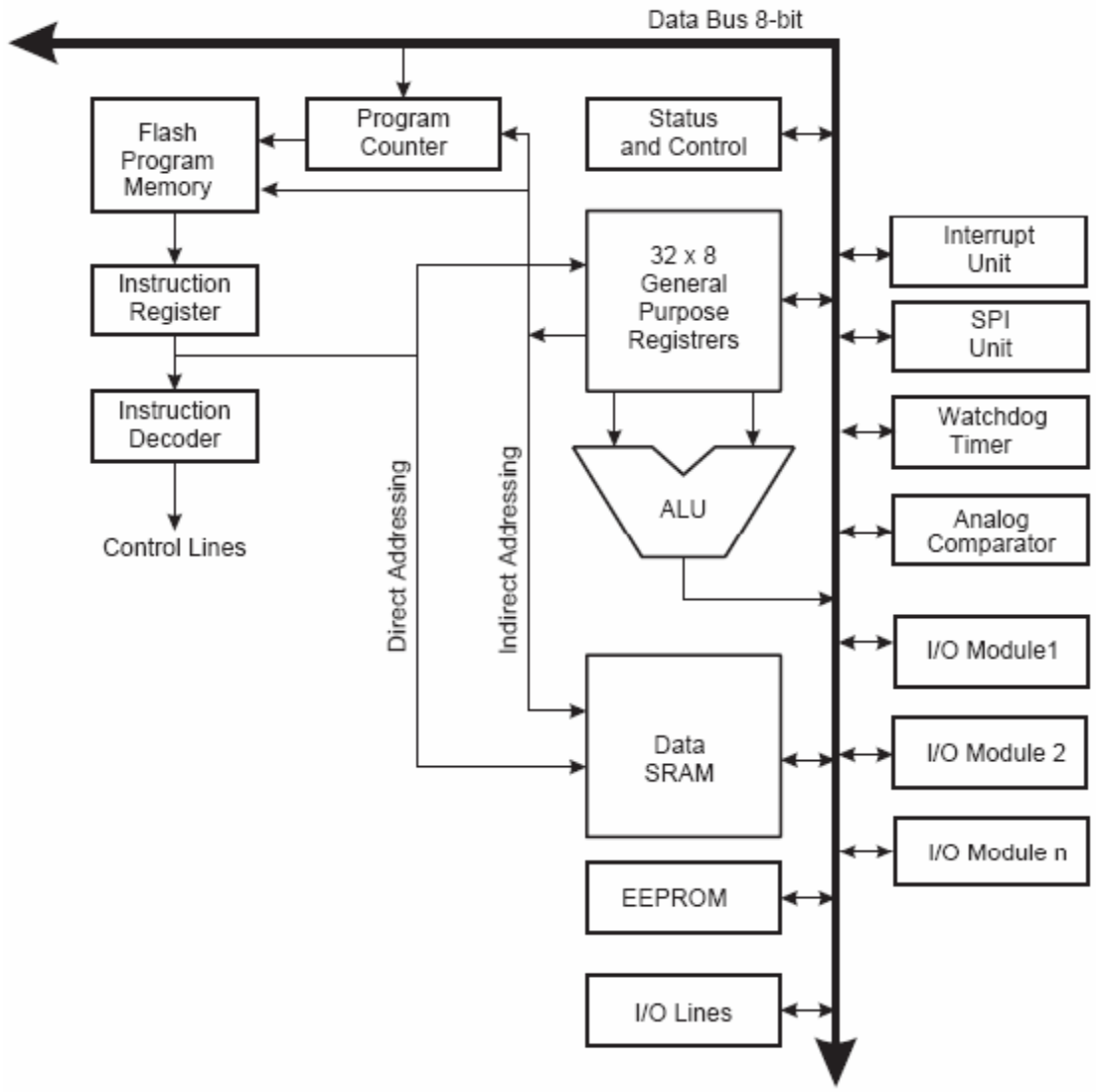
Vi điều khiển AVR do hãng Atmel sản xuất được gới thiệu lần đầu năm 1996. AVR có rất nhiều dòng khác nhau bao gồm dòng Tiny AVR (như AT tiny 13, ATtiny 22…) có kích thước bộ nhớ nhỏ, ít bộ phận ngoại vi, rồi đến dòng AVR (chẳn hạn AT90S8535, AT90S8515,…) có kích thước bộ nhớ vào loại trung bình và mạnh hơn là dòng Mega (như ATmega32, ATmega128,…) với bộ nhớ có kích thước vài Kbyte đến vài trăm Kb cùng với các bộ ngoại vi đa dạng được tích hợp trên chip, cũng có dòng tích hợp cả bộ LCD trên chip (dòng LCD AVR). Tốc độ của dòng Mega cũng cao hơn so với các dòng khác. Sự khác nhau cơ bản giữa các dòng chình là cấu trúc ngoại vi, còn nhân thì vẫn như nhau.

Hình 4.2 Các dòng AVR

Những đặc trưng chính bao gồm

- ROM : 128 Kbytes
- SRAM: 4Kbytes
- EEPROM : 4Kbytes
- 64 thanh ghi I/O
- 160 thanh ghi vào ra mở rộng
- 32 thanh ghi đa mục đích.
- bộ định thời 8 bit (0,2).
- bộ định thời 16 bit (1,3).
- Bộ định thời watchdog
- Bộ dao động nội RC tần số 1 MHz, 2 MHz, 4 MHz, 8 MHz
- ADC 8 kênh với độ phân giải 10 bit (Ở dòng Xmega lên tới 12 bit)
- 2 kênh PWM 8 bit
- 6 kênh PWM có thể lập trình thay đổi độ phân giải từ 2 tới 16 bit
- Bộ so sánh tương tự có thể lựa chọn ngõ vào
- Hai khối USART lập trình được
- Khối truyền nhận nối tiếp SPI
- Khối giao tiếp nối tiếp 2 dây TWI
- Hỗ trợ boot loader
- 6 chế độ tiết kiệm năng lượng
- Lựa chọn tần số hoạt động bằng phần mềm
- Đóng gói 64 chân kiểu TQFP.
- Tần số tối đa 16MHz
- Điện thế : 4.5v - 5.5v

4.3.2 Kiến trúc họ AVR



Hình 4.3 Sơ đồ khối AVR

Bộ nhớ Memory

Bộ nhớ vi điều khiển AVR có cấu trúc Harvard là cấu trúc có đường Bus riêng cho bộ nhớ chương trình và bộ nhớ dữ liệu. Bộ nhớ AVR được chia làm 2 phần chính: Bộ nhớ chương trình (program memory) và bộ nhớ dữ liệu (Data memory).

Bộ Nhớ Chương Trình : Bộ nhớ chương trình của AVR là bộ nhớ Flash có dung lượng 128 K bytes. Bộ nhớ chương trình có độ rộng bus là 16 bit. Những địa chỉ đầu tiên của bộ nhớ chương trình được dùng cho bảng véc tơ ngắt. Cần để ý là ở vi điều khiển ATmega128 bộ nhớ chương trình còn có thể được chia làm 2 phần : phần boot loader (Boot loader program section) và phần ứng dụng (Application program section).

Phần boot loader chứa chương trình boot loader. Chương trình Boot loader là một phần mềm nhỏ nạp trong vi điều khiển và được chạy lúc khởi động. Phần mềm này có thể tải vào trong vi điều khiển chương trình của người sử dụng và sau đó thực thi chương trình này. Mỗi khi reset vi điều khiển CPU sẽ nhảy tới thực thi chương trình boot loader trước, chương trình boot loader sẽ dò xem có chương trình nào cần nạp vào vi điều khiển hay không, nếu có chương trình cần nạp, boot loader sẽ nạp chương trình vào vùng nhớ ứng dụng (Application program section), rồi thực thi chương trình này. Ngược lại, boot loader sẽ chuyển tới chương trình ứng dụng có sẵn trong vùng nhớ ứng dụng để thực thi chương trình này.

Phần ứng dụng (Application program section) là vùng nhớ chứa chương trình ứng dụng của người dùng. Kích thước của phần boot loader và phần ứng dụng có thể tùy chọn. Hình trên thể hiện cấu trúc bộ nhớ chương trình có sử dụng và không sử dụng boot loader, khi sử dụng phần boot loader ta thấy 4 word đầu tiên thay vì chỉ thị cho CPU chuyển tới chương trình ứng dụng của người dùng (là chương trình có nhãn start) thì chỉ thị CPU nhảy tới phần chương trình boot loader để thực hiện trước, rồi mới quay trở lại thực hiện chương trình ứng dụng.

Bộ Nhớ Dữ Liệu : Bộ nhớ dữ liệu của AVR chia làm 2 phần chính là bộ nhớ SRAM và bộ nhớ EEPROM. Tuy cùng là bộ nhớ dữ liệu nhưng hai bộ nhớ này lại tách biệt nhau và được đánh địa chỉ riêng.

Bộ nhớ SRAM có dụng lượng 4 K bytes, Bộ nhớ SRAM có hai chế độ hoạt động là chế độ thông thường và chế độ tương thích với ATmega103, muốn thiết lập bộ nhớ SRAM hoạt động theo chế độ nào ta sử dụng bit cầu chì M103C.

Bộ nhớ EEPROM : Đây là bộ nhớ dữ liệu có thể ghi xóa ngay trong lúc vi điều khiển đang hoạt động và không bị mất dữ liệu khi nguồn điện cung cấp bị cắt. Có thể ví bộ nhớ dữ liệu EEPROM giống như là ổ cứng (Hard disk) của máy vi tính. Với vi điều khiển ATmega128, bộ nhớ EEPROM có kích thước là 4 Kbyte. EEPROM được xem như là một bộ nhớ vào ra được đánh địa chỉ độc lập với SRAM, điều này có nghĩa là ta cần sử dụng các lệnh in, out … khi muốn truy xuất tới EEPROM.

4.3.3 Tập lệnh của AVR

Tập lệnh được chia thành các nhóm sau:

Các lệnh toán học và logic: Một số lệnh đáng chú ý trong nhóm này là cộng (ADD), trừ (SUB), nhân (MUL), chia (FMUL), xóa (CLR), tăng (INC), giảm (DEC).

Lệnh rẽ nhánh: Gồm có các lệnh nhảy, lệnh gọi chương trình con, lệnh trở về, lệnh so sánh, lệnh rẽ nhánh.

Lệnh di chuyển dữ liệu: Gồm có các lệnh sao chép dữ liệu (MOV), xuất nhập dữ liệu, PUSH, POP.

Các lệnh thao tác BIT: SBI, CBI, LSL, …

Các lệnh điều khiển MCU: NOP, SLEEP, WDR

4.3.4 Sự thực thi

Cổng vào ra

Cổng vào ra là một trong số các phương tiện để vi điều khiển giao tiếp với các thiết bị ngoại vi. ATmega128 có cả thảy 7 cổng (port) vào ra 8 bit là : PortA, PortB, PortC, PortD, PortE, PortF, PortG, tương ứng với 56 đường vào ra. Các cổng vào ra của AVR là cổng vào ra hai chiều có thể định hướng, tức có thể chọn hướng của cổng là hướng vào (input) hay hướng ra (output). Tất các các cổng vào ra của AVR điều có tính năng Đọc – Chỉnh sửa – Ghi (Read – Modify – write) khi sử dụng chúng như là các cổng vào ra số thông thường. Điều này có nghĩa là khi ta thay đổi hướng của một chân nào đó thì nó không làm ảnh hưởng tới hướng của các chân khác. Tất cả các chân của các cổng (port) điều có điện trở kéo lên (pull-up) riêng, ta có thể cho phép hay không cho phép điện trở kéo lên này hoạt động.

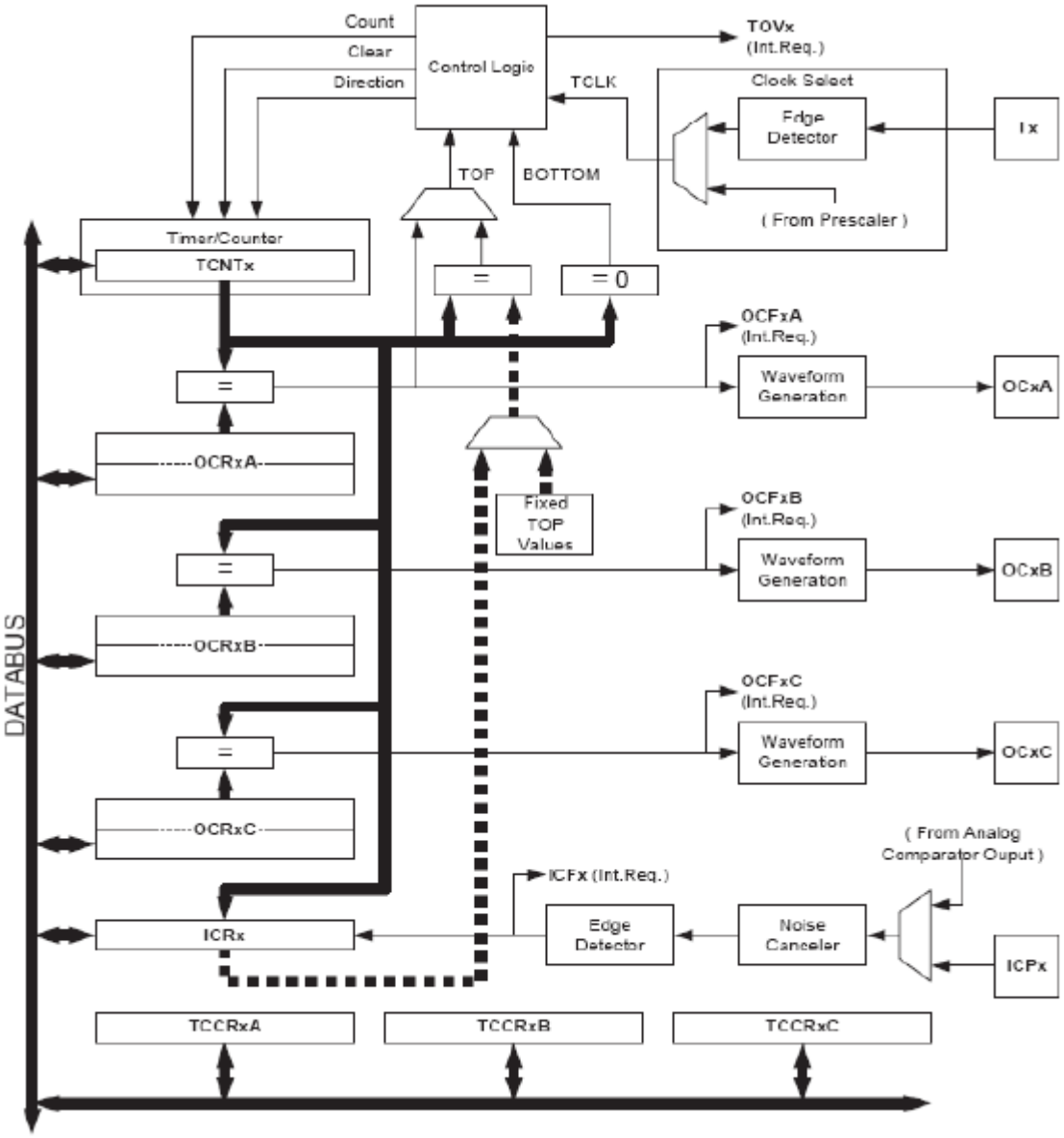
Điện trở kéo lên là một điện trở được dùng khi thiết kế các mạch điện tử logic. Nó có một đầu được nối với nguồn điện áp dương (thường là Vcc hoặc Vdd) và đầu còn lại được nối với tín hiệu lối vào/ra của một mạch logic chức năng. Điện trở kéo lên có thể được lắp đặt tại các lối vào của các khối mạch logic để thiết lập mức logic lối vào của khối mạch khi không có thiết bị ngoài nối với lối vào. Điện trở kéo lên cũng có thể được lắp đặt tại các giao diện giữa hai khối mạch logic không cùng loại logic, đặc biệt là khi hai khối mạch này được cấp nguồn khác nhau. Ngoài ra, điện trở kéo lên còn được lắp đặt tại lối ra của khối mạch khi lối ra không thể nối nguồn để tạo dòng, ví dụ các linh kiện logic TTL có cực góp hở. Đối với họ logic lưỡng cực với nguồn nuôi 5 Vdc thì giá trị của điện trở kéo lên thường nằm trong

khoảng 1000 đến 5000 Ohm, tùy theo yêu cầu cấp dòng trên toàn giải hoạt động của mạch. Với lôgíc CMOS và lôgíc MOS chúng ta có thể sử dụng các điện trở có giá trị lớn hơn nhiều, thường từ vài ngàn đến một triệu Ohm do dòng rò rỉ cần thiết ở lối vào là rất nhỏ. Trong việc thiết kế các vi mạch ứng dụng, nếu một IC có ngõ ra loại cực thu để hở giao tiếp với nhiều IC khác thì giá trị của điện trở kéo lên sẽ tương đối nhỏ (khoảng vài trăm Ohm). Bởi vì lúc này hệ số fanout lớn dẫn đến dòng ngõ ra của IC phải lớn để đủ cung cấp cho các ngõ vào của các IC khác, nếu không vi mạch sẽ hoạt động chập chờn hoặc có thể không hoạt động.

Mỗi một cổng vào ra của vi điều khiển được liên kết với 3 thanh ghi : PORTx, DDRx, PINx. (ở đây x là để thay thế cho A, B,…G). Ba thanh ghi này sẽ được phối hợp với nhau để điều khiển hoạt động của cổng, chẳn hạn thiết lập cổng thành lối vào có sử dụng điện trở pull-up, ..v.v.. .

Bộ định thời

Ta có 4 bộ định thời , bộ định thời 1 và 3 là bộ định thời 16 bit, bộ định thời 0 và 2 là bộ định thời 8 bit. Dưới đây là mô tả chi tiết của 4 bộ định thời.



Hình 4.4 Sơ đồ khối bộ định thời

Bộ định thời 1 và 3 là bộ định thời 16 bit, bộ định thời 1 sử dụng 13 thanh ghi liên quan, còn bộ định thời 3 sử dụng 11 thanh ghi liên quan với nhiều chế độ thực thi khác nhau.

Một đểm cần để ý là trong các thanh ghi liên quan tới bộ định thời 1 và 3 thì có nhiều thanh ghi được chia sẽ cho cả hai bộ định thời, chẳn hạn thanh ghi ETIPR có bít cuối là OCF1C được dùng cho bộ định thời 1, các bit còn lại là dùng cho bộ định thời 3. Thậm chí có những thanh ghi chia sẽ cho bộ định thời 0 hoặc 2, chẳn hạn thanh ghi TIMSK có hai bit cuối dùng cho bộ định thời 2, hai bit đầu dùng cho bộ định thời 0, các bit còn lại dùng cho bộ định thời 1. Để tìm hiểu về bộ định thời 1 (3) ta cần nắm vững các thanh ghi liên quan tới bộ định thời 1(3) và các chế độ hoạt động của bộ định thời.

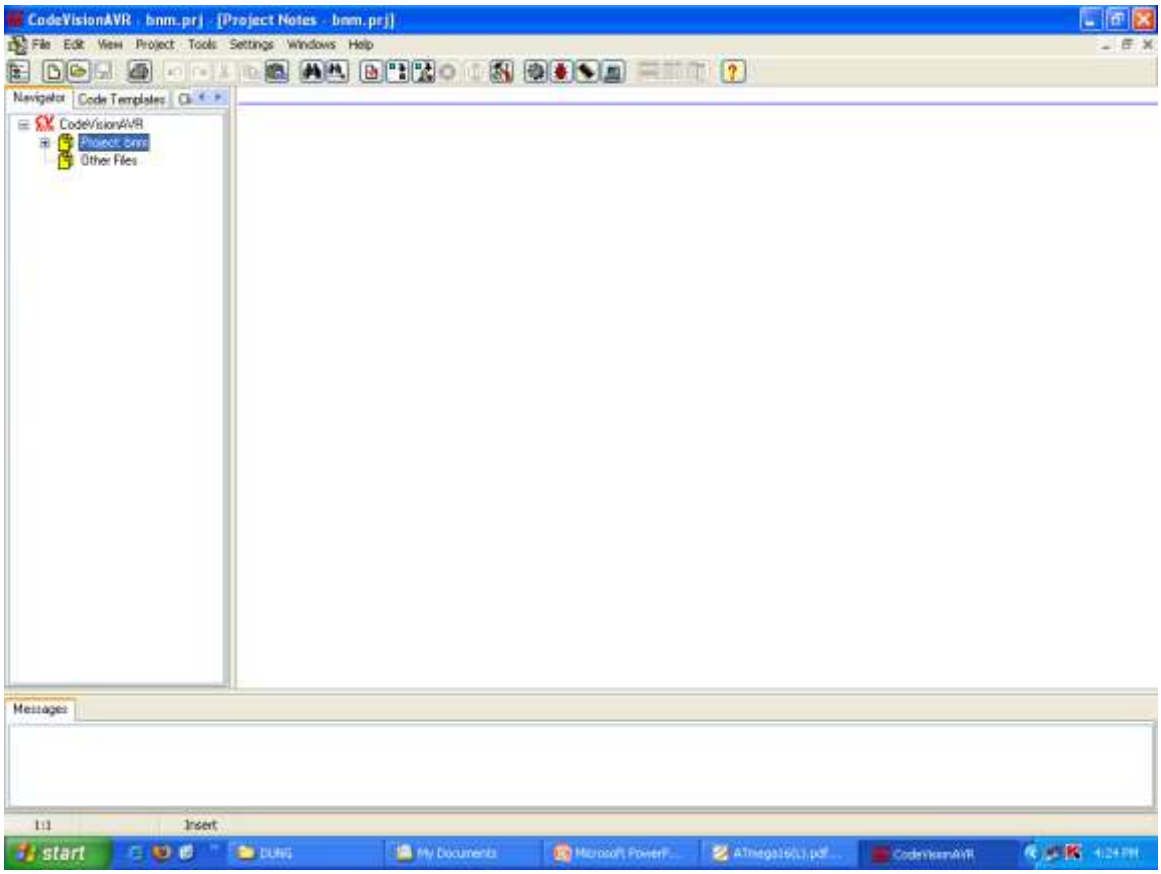
4.3.5 Thiết kế ứng dụng

Trong phần này này ta sử dụng phần mềm thông dụng CodeVision để lập trình cổng vào ra cho Atmega16 (Thuộc họ AVR) bằng cách tác động vào thanh ghi PORTxx và DDRxx.

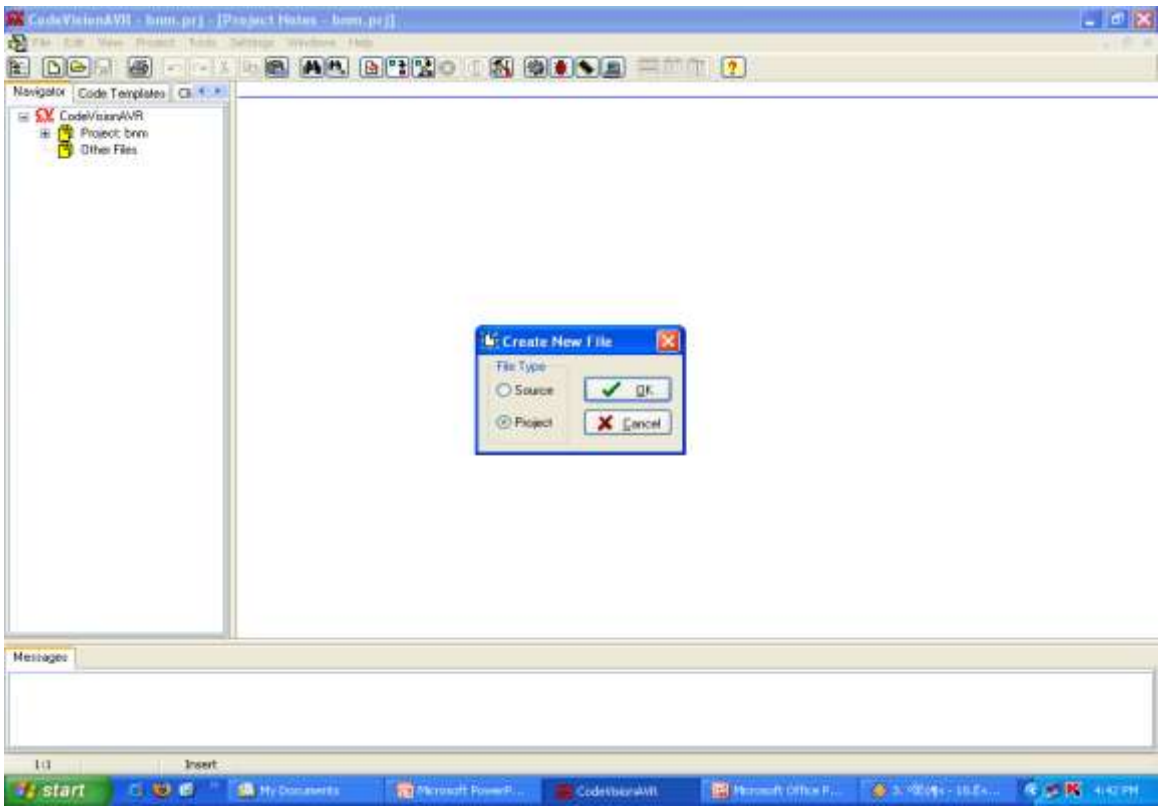
DDRxx : để điều khiển các hướng dữ liệu các chân của cổng. Khi DDRxx=0 thì dùng làm cổng vào, ngược lại, khi DDRxx=1 thì dùng làm cổng ra.

PORTxx: truy cập tại các địa chỉ xuất nhập của PORTx

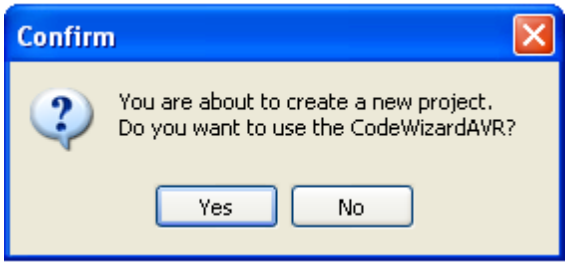
- Chạy chương trình CodeVision, chọn File -> New -> Project -> OK



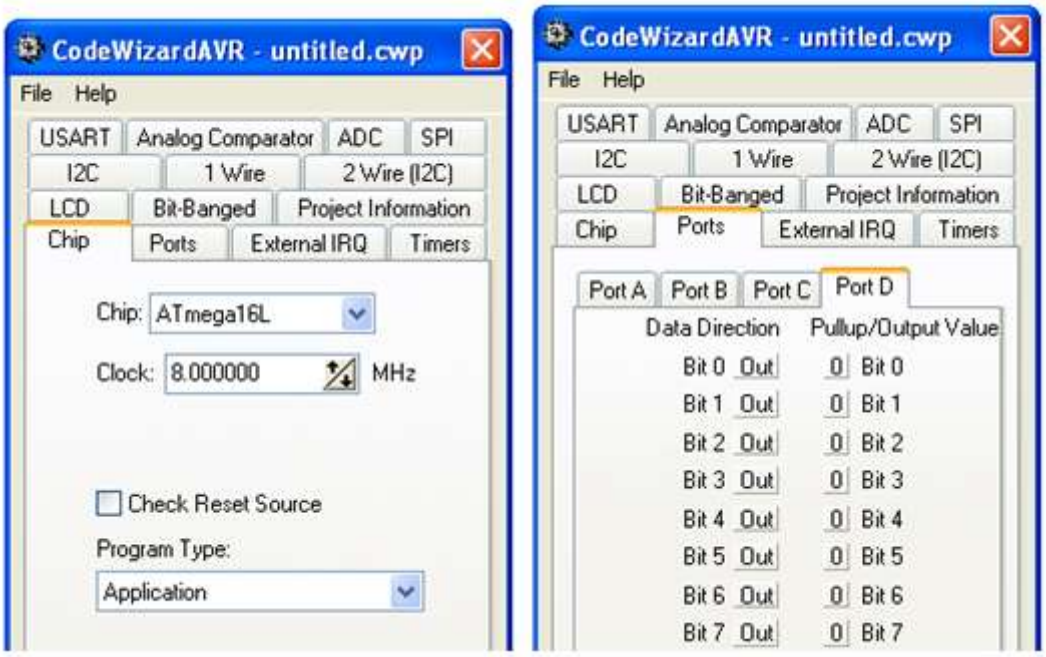
- Tiếp theo, ta chọn Project:

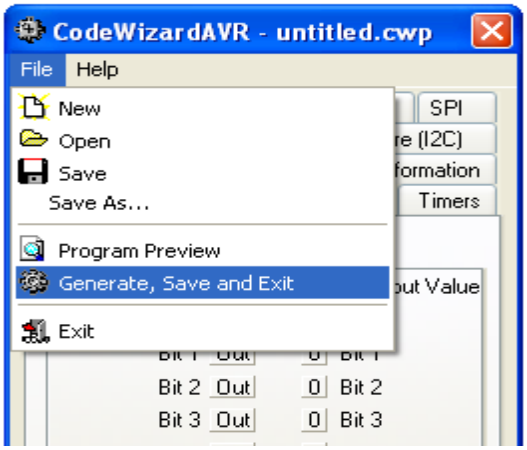


- Chọn sử dụng Code Wizard

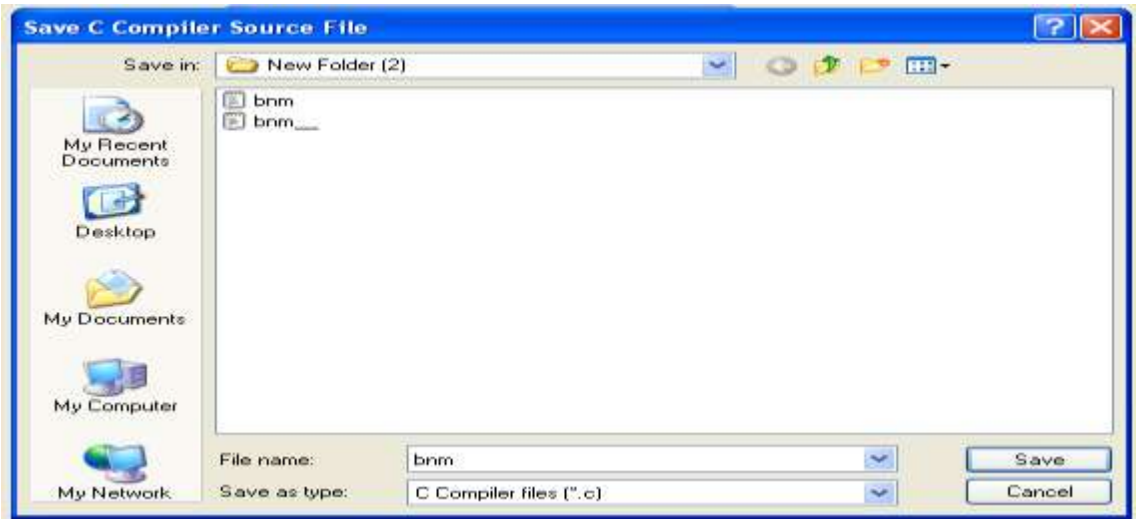


- Cửa sổ Code Vision Wizard

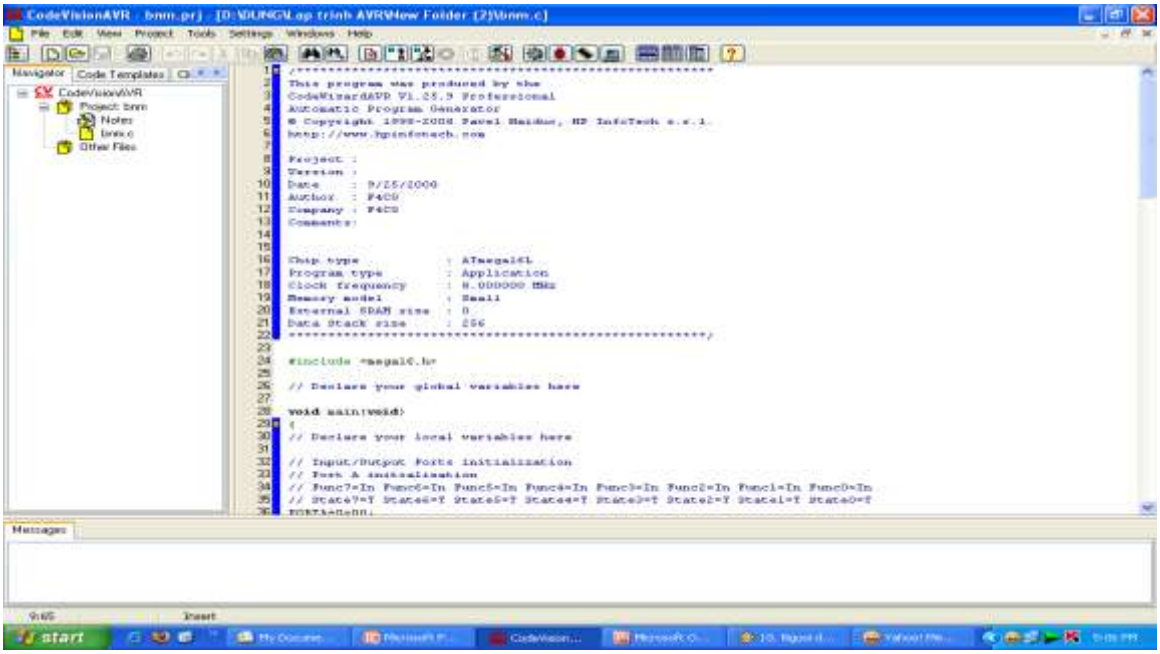




- Lưu lại các file



- Soạn thảo mã lệnh:



4.4 Họ vi xử lý ARM

4.4.1 Tổng quan

Ngày 26/4/1985, mẫu sản phẩm ARM đầu tiên sản xuất tại công ty kĩ thuật VLSI, SanJose, bang Califonia được chuyển tới trung tâm máy tính Acorn ở Cambridge, Anh Quốc. Một vài giờ sau, chương trình thử nghiệm đầu tiên thành công. Nửa thập niên sau đó, ARM được phát triển rất nhanh chóng để làm nhân máy tính để bàn của Acorn, nền tảng cho các máy tính hỗ trợ giáo dục ở Anh. Trong thập niên 1990, dưới sự phát triển của Acorn Limited, ARM đã thành một thương hiệu đứng đầu thế giới về các ứng dụng sản phẩm nhúng đòi hỏi tính năng cao, sử dụng năng lượng ít và giá thành thấp.

Chính nhờ sự nổi trội về thị phần đã thúc đẩy ARM liên tục được phát triển và cho ra nhiều phiên bản mới. Những thành công quan trọng trong việc phát triển ARM ở thập niên sau này:

- Giới thiệu ý tưởng về định dạng các chỉ lệnh được nén lại (thumb) cho phép tiết kiệm năng lượng và giá thành ở những hệ thống nhỏ.
- Giới thiệu họ điều khiển ARM9, ARM10 và 'Strong ARM'
- Phát triển môi trường làm việc ảo của ARM trên PC.
- Các ứng dụng cho hệ thống nhúng dựa trên nhân xử lý ARM ngày càng trở nên rộng rãi.

Hầu hết các nguyên lý của hệ thống trên chip (Systems on chip-SoC) và cách thiết kế bộ xử lý hiện đại được sử dụng trong ARM, ARM còn đưa ra một số khái niệm mới <như giải nén động các dòng lệnh>. Việc sử dụng 3 trạng thái nhận lệnh-giải mã-thực thi trong mỗi chu kì máy mang tính quy phạm để thiết kế các hệ thống xử lý thực. Do đó, nhân xử lý ARM được sử dụng rộng rãi trong các hệ thống phức tạp.

4.4.2 Kiến trúc họ

ARM lúc đầu được đặt tên theo công ty Acorn. ARM=Acorn RISC Machine (dịch môm na là chiếc máy sử dụng tập lệnh đơn giản của công ty Acorn). Sau này, do có thêm nhiều công ty cùng phát triển và một số lý do khác, người ta thống nhất gọi ARM=Advance RISC Machine.

Cấu trúc cơ bản:

- Cấu trúc load-store
- Chỉ lệnh có chiều dài cố định <32bit>
- Cấu trúc chỉ lệnh có 3 địa chỉ.

Thay vì chỉ dùng 1 chu kì xung nhịp cho tất cả các chỉ lệnh, ARM thiết kế để sao cho tối giản số chu kì xung nhịp cho một chỉ lệnh, do đó tăng được sự phức tạp cho các chỉ lệnh đơn lẻ. Cũng như hầu hết các bộ xử lý dùng tập lênh RISC khác, ARM cũng sử dụng cấu trúc load-store. Điều đó có nghĩa là: tất cả các chỉ lệnh <cộng, trừ…> đều được thực hiện trên thanh ghi. Chỉ có lệnh copy giá trị từ bộ nhớ vào thanh ghi<load> hoặc chép lại giá trị từ thanh ghi vào bộ nhớ<store> mới có ảnh hưởng tới bộ nhớ. Các bộ xử lý CISC cho phép giá trị trên thanh ghi có thể cộng với giá trị trong bộ nhớ, đôi khi còn cho phép giá trị trên bộ nhớ có thể cộng với giá trị trên thanh ghi. ARM không hỗ trợ cấu trúc lệnh dạng 'từ bộ nhớ đến bộ nhớ'. Vì thế, tất cả các lệnh của ARM có thể thuộc 1 trong 3 loại sau:

- Chỉ lệnh xử lý dữ liệu: chỉ thay đổi giá trị trên thanh ghi.
- Chỉ lệnh truyền dữ liệu: copy giá trị từ thanh ghi vào bộ nhớ và chép giá trị từ bộ nhớ vào thanh ghi.<load-store>
- Chỉ lệnh điều khiển dòng lệnh: Bình thường, ta thực thi các chỉ lệnh chứa trong một vùng nhớ liên tiếp, chỉ lệnh điều khiển dòng lệnh cho phép chuyển sang các địa chỉ khác nhau khi thực thi lệnh, tới những nhánh cố định, <lệnh rẽ nhánh>

hoặc là lưu và trở lại địa chỉ để phục hồi chuỗi lệnh ban đầu <chỉ lệnh rẽ nhánh và kết nối> hay là chèn lên vùng code của hệ thống <gọi giám sát-ngắt phần mềm>.

4.4.3 Tập lệnh

Tất cả lệnh của ARM đều là 32bit:

- Có cấu trúc dạng load-store.
- Cấu trúc lệnh định dạng 3 địa chỉ (nghĩa là địa chỉ của 2 toán hạng nguồn và toán hạng đích đều là các địa chỉ riêng biệt)
- Mỗi chỉ lệnh thực thi một điều kiện.
- Có cả chỉ lệnh load-store nhiều thanh ghi đồng thời.
- Có khả năng dịch bit kết hợp với thực thi lệnh ALU trong chỉ 1 chu kì máy.

Các lệnh của ARM có thể chia thành các loại sau:

Cấu trúc chỉ lệnh có 4 địa chỉ

Dạng lệnh là:

Tên lệnh Địa chỉ 1 Địa chỉ 2 Địa chỉ Đích Địa chỉ kế tiếp

Ví dụ:

ADD d, s1, s2, next_i ;d := s1 +s2

Cấu trúc chỉ lệnh có 3 địa chỉ:

Dạng lệnh:

Tên lệnh Địa chỉ 1 Địa chỉ 2 Địa chỉ Đích

Ví dụ:

ADD d, s1, s2 ; d := s1 + s2

Cấu trúc chỉ lệnh có 2 địa chỉ:

Dạng lệnh:

Tên lệnh Địa chỉ 1 Địa chỉ Đích

Ví dụ:

ADD d, s1 ; d := d + s1

Cấu trúc chỉ lệnh có 1 địa chỉ:

Dạng lệnh:

Tên lệnh Địa chỉ 1

Ví dụ:

ADD s1 ; accumulator := accumulator + s1

Cấu trúc chỉ lệnh không truy cập địa chỉ:

Dạng lệnh:

Tên lệnh

Ví dụ

ADD ; top_of_stack := top_of_stack +next_on_stack

4.4.4 Sự thực thi

Cũng như hầu hết các bộ xử lý dùng tập lệnh RISC khác, ARM cũng sử dụng cấu trúc load-store. Điều đó có nghĩa là: tất cả các chỉ lệnh <cộng, trừ…> đều được thực hiện trên thanh ghi. Chỉ có lệnh copy giá trị từ bộ nhớ vào thanh ghi<load> hoặc chép lại giá trị từ thanh ghi vào bộ nhớ<store> mới có ảnh hưởng tới bộ nhớ. Các bộ xử lý CISC cho phép giá trị trên thanh ghi có thể cộng với giá trị trong bộ nhớ, đôi khi còn cho phép giá trị trên bộ nhớ có thể cộng với giá trị trên thanh ghi. ARM không hỗ trợ cấu trúc lệnh dạng 'từ bộ nhớ đến bộ nhớ'. Vì thế, tất cả các lệnh của ARM có thể thuộc 1 trong 3 loại sau:

- Chỉ lệnh xử lý dữ liệu: chỉ thay đổi giá trị trên thanh ghi.
- Chỉ lệnh truyền dữ liệu: copy giá trị từ thanh ghi vào bộ nhớ và chép giá trị từ bộ nhớ vào thanh ghi.<load-store>
- Chỉ lệnh điều khiển dòng lệnh: Bình thường, ta thực thi các chỉ lệnh chứa trong một vùng nhớ liên tiếp, chỉ lệnh điều khiển dòng lệnh cho phép chuyển sang các địa chỉ khác nhau khi thực thi lệnh, tới những nhánh cố định, <lệnh rẽ nhánh> hoặc là lưu và trở lại địa chỉ để phục hồi chuỗi lệnh ban đầu <chỉ lệnh rẽ nhánh và kết nối> hay là đè lên vùng code của hệ thống <gọi giám sát-ngắt phần mềm>.

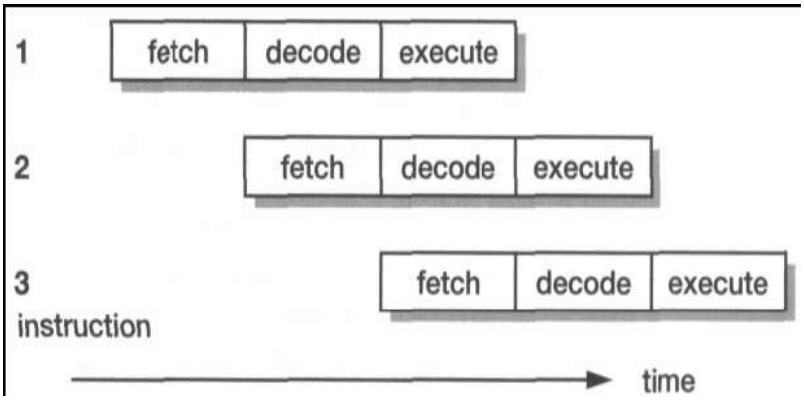
4.4.5 Thiết kế ứng dụng

Trong phần này chúng ta sẽ tìm hiểu phương pháp lập trình hợp ngữ với ARM.

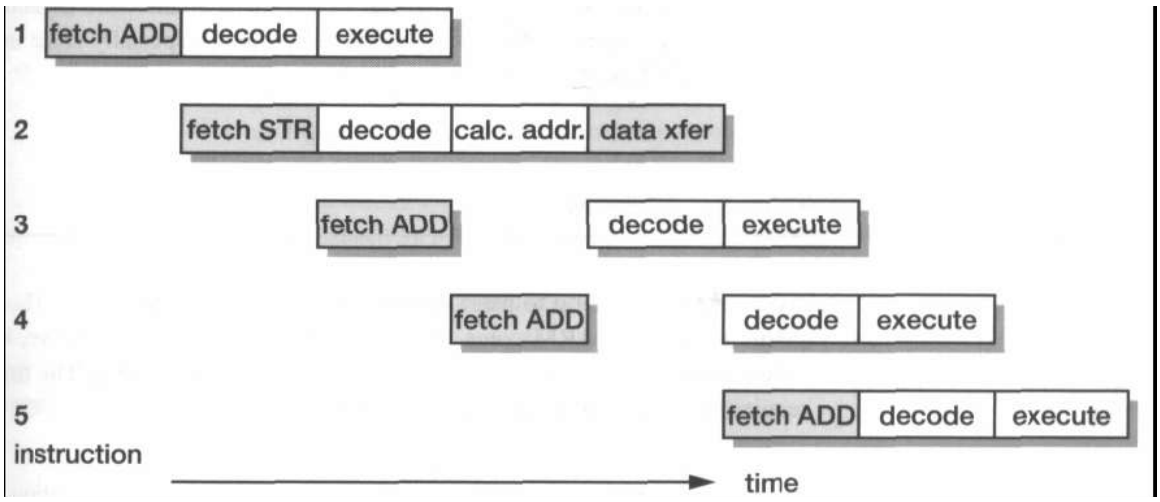
Xét chương trình sau:

```
AREA HelloW,CODE,READONLY        ;Khai bao vung code
SWI_SwiteC EQU &0                ;Ki tu xuat o R0
SWI_Exit EQU &11                 ;Ket thuc chuong trinh
ENTRY                            ;Diem bao hieu vao chuong trinh
START ADR R1,TEXT                ;R1 chi den vung dia chi cua TEXT
LOOP LDRB R0,[R1],#1             ;R0:=[R1];R1:=R1+1
CMP R0,#0                        ;R0 chi toi gia tri cuoi hay chua
SWINE SWI_WriteC                 ;Neu chua ket thuc in
BNE LOOP                         ;thi quay nguoc lai vong lap
SWI SWI_Exit                     ;Quay  ve lai chuong trinh quan ly
TEXT ="Hello World",&0a,&0d,0    ;Khai bao doan Text
END                              ;Chuong trinh ket thuc
```

Cách tổ chức và thực thi tập lệnh của ARM: Cách tổ chức của nhân ARM không thay đổi nhiều trong khoảng 1983-1995:đến ARM7-sử dụng dòng chảy lệnh sử dụng 3 tác vụ. Từ 1995 trở về sau, đã xuất hiện một vài nhân ARM mới được giới thiệu có dòng chảy lệnh sử dụng 5 tác vụ.



Hình 4.5 Chỉ lệnh một chu kì máy sử dụng dòng chảy lệnh có 3 tác vụ



Hình 4.6 Dòng chảy lệnh 3 tác vụ

Thời gian để bộ xử lý thực thi một chương trình là:

$$T_{prog} = \frac{N_{inst}\, x\, CPI}{f_{clk}}$$

Trong đó CPI là số xung nhịp trung bình cần cho mỗi chỉ lệnh, Ninst là số chỉ lệnh thực thi một chương trình<là cố định>, fclk là tần số xung nhịp. Với công thức trên, ta có 2 cách để giảm thời gian thực thi một chương trình:

+ Tăng tần số xung nhịp: điều này đòi hỏi trạng thái của mỗi tác vụ trong dòng chảy lệnh là đơn giản, và, do đó, số tác vụ sẽ tăng thêm.

+ Giảm CPI: điều này đòi hỏi mỗi chỉ lệnh cần nhiều dòng chảy lệnh hơn với tác vụ không đổi, hoặc các tác vụ cần đơn giản hơn, hoặc kết hợp cả 2 lại với nhau.

ARM đưa ra cấu trúc mỗi dòng chảy lệnh có 5 tác vụ, với cách mô phỏng tựa như cấu trúc von Neumann, với vùng nhớ dữ liệu và chương trình riêng biệt. Từ cấu trúc lệnh có 3 tác vụ được chia nhỏ lại thành 5 tác vụ cũng làm cho mỗi chu kì xung nhịp sẽ thực hiện một công việc đơn giản hơn ở mỗi trạm, cho phép có thể tăng chu kì xung nhịp của hệ thống. Sự tách rời bộ nhớ chương trình và bộ nhớ dữ liệu <cache chứa các chỉ lệnh I-cache và cache chứa dữ liệu D-cache là tách rời nhau> cũng cho phép giảm đáng kể tài nguyên chiếm của mỗi chỉ lệnh trong một chu kì máy.

Câu hỏi cuối chương

1. Trình bày các đặc điểm của họ vi xử lý AT89C
2. Nêu kiến trúc chung của họ vi xử lý AT89C
3. Trình bày về các lệnh logic của AT89C
4. Trình bày các thanh ghi của Timer trong vi xử lý AT89C
5. Trình bày các chế độ hoạt động của Timer trong AT89C
6. Trình bày về bộ nhớ và các cổng trong họ vi xử lý AVR
7. Nêu các bước để lập trình cho AVR với CodeVision
8. Trình bày các đặc điểm của các lệnh trong vi xử lý AVR
9. Phân tích sự khác nhau giữa dòng chảy lệnh có 3 tác vụ và dòng chảy lệnh có 5 tác vụ

ĐỀ THI THAM KHẢO

Đề số 1

Câu 1: Trình bày các phương pháp biểu diễn kiến trúc của hệ thống nhúng.

Câu 2: Phân tích đặc điểm, chức năng của hệ thống bus địa chỉ trong hệ thống nhúng.

Câu 3: Trình bày về trình điều khiển ngắt.

Câu 4: Nêu các bước lập trình cho AVR với CodeVision

Đề số 2

Câu 1: Thế nào là Soft Real Time System, cho ví dụ.

Câu 2: Nêu cấu tạo, đặc điểm các loại bộ nhớ RAM.

Câu 3: Trình bày những đặc điểm của hệ điều hành trong các hệ thống nhúng.

Câu 4: Trình bày về các chế độ hoạt động của Timer trong vi xử lý AT89C

Đề số 3

Câu 1: Thế nào là Hard Real Time System, cho ví dụ.

Câu 2: Nêu cấu tạo, đặc điểm các loại bộ nhớ ROM.

Câu 3: Nêu và phân tích các điều kiện cần của RMS.

Câu 4: Nêu kiến trúc chung của họ vi xử lý AT89C

Đề số 4

Câu 1: Trình bày về mô hình phát triển hệ thống nhúng Spiral.

Câu 2: Nêu các cơ chế quản lý bộ nhớ với hệ thống nhúng.

Câu 3: Thế nào là tiến trình, trình bày về quản lý tiến trình trong hệ thống.

Câu 4: Nêu đặc điểm của tập lệnh của vi xử lý ARM.

Đề số 5

Câu 1: Trình bày về mô hình phát triển hệ thống nhúng Water Fall và các ưu khuyết điểm của nó so với các phương pháp truyền thống.

Câu 2: Nêu những ưu điểm của phương pháp vào ra song song.

Câu 3: So sánh các hệ thống đơn tiến trình và các hệ thống đa tiến trình.

Câu 4: Trình bày về hoạt động của các cổng trong vi xử lý AVR.

Đề số 6

Câu 1: Tại sao ta sử dụng phương pháp phân lớp để biểu diễn kiến trúc hệ thống nhúng?

Câu 2: Trình bày những đặc điểm phần cứng của một hệ thống nhúng.

Câu 3: Thế nào là MiddleWare software, Application software? Cho ví dụ.

Câu 4: Nêu sự thực thi của các lệnh trong vi xử lý ARM.

GỢI Ý ĐÁP ÁN

Đề số 1

Câu 1: Trình bày các cấu trúc có thể được dùng để biểu diễn kiến trúc của hệ thống nhúng.

Gợi ý

- Module

- Component and Connector
- Allocation

Câu 2: Phân tích đặc điểm, chức năng của hệ thống bus địa chỉ trong hệ thống nhúng.

Gợi ý

- Vị trí
- Độ rộng bus
- Cấu tạo
- Nhiệm vụ

Câu 3: Trình bày về trình điều khiển ngắt.

Gợi ý

- Ngắt Level Triggered
- Ngắt Edge Triggered
- Các chức năng chính của trình điều khiển

Câu 4: Nêu các bước lập trình cho AVR với CodeVision

Gợi ý

Trình bày các bước làm việc với Code Vision Wizard trong Code Vision

Đề số 2

Câu 1: Thế nào là Soft Real Time System, cho ví dụ.

Gợi ý:

Trình bày về các hệ thống thời gian thực trong đó sai khác về thời gian không gây hậu quả lớn về tài sản, con người.

Câu 2: Nêu cấu tạo, đặc điểm các loại bộ nhớ RAM.

Gợi ý

- Trình bày về các loại SDRAM, DRAM …
- Cấu tạo các ô nhớ và đặc điểm
- Phương pháp truy cập đến các phần tử nhớ

Câu 3: Trình bày những đặc điểm của hệ điều hành trong các hệ thống nhúng.

Gợi ý

Câu 4: Trình bày về các chế độ hoạt động của Timer trong vi xử lý AT89C

Đề số 3

Câu 1: Thế nào là Hard Real Time System, cho ví dụ.

Gợi ý

Trình bày những hệ thống mà trong đó việc sai khác về thời gian sẽ dẫn đến những hậu quả nghiêm trọng về tài sản và người.

Câu 2: Nêu cấu tạo, đặc điểm các loại bộ nhớ ROM.

Gợi ý

- Trình bày về các loại ROM …
- Cấu tạo các ô nhớ và đặc điểm
- Phương pháp truy cập đến các phần tử nhớ

Câu 3: Nêu và phân tích các điều kiện cần của RMS.

Gợi ý

- Trình bày đơn sử dụng
- Trình bày về đa sử dụng

Câu 4: Nêu kiến trúc chung của họ vi xử lý AT89C

Gợi ý

- Sơ đồ khối
- Mô tả các khối
- Mô tả các chân

Đề số 4

Câu 1: Trình bày về mô hình phát triển hệ thống nhúng Spiral.

Gợi ý

- Sơ đồ khối
- Trình bày các phase của mô hình
- So sánh với mô hình warterfall

Câu 2: Nêu các cơ chế quản lý bộ nhớ với hệ thống nhúng.

Gợi ý

- Trình bày về phân cấp bộ nhớ trong hệ thống nhúng
- Trình bày về Memory Management

Câu 3: Thế nào là tiến trình, trình bày về quản lý tiến trình trong hệ thống.

Gợi ý

- Định nghĩa chương trình, tiến trình
- Đơn tiến trình
- Đa tiến trình

Câu 4: Nêu đặc điểm của tập lệnh của vi xử lý ARM.

Gợi ý

- Kích thước lệnh
- Dạng lệnh
- Kiểu thực thi

Đề số 5

Câu 1: Trình bày về mô hình phát triển hệ thống nhúng Water Fall và các ưu khuyết điểm của nó so với các phương pháp truyền thống.

Gợi ý

- Sơ đồ khối
- Trình bày các phase của mô hình
- So sánh với mô hình Spiral

Câu 2: Nêu những ưu điểm của phương pháp vào ra song song.

Gợi ý

- Tốc độ
- Cấu tạo dây
- Kết nối với thiết bị ngoại vi

Câu 3: So sánh các hệ thống đơn tiến trình và các hệ thống đa tiến trình.

Gợi ý

- Độ phức tạp của thuật toán quản lý
- Hiệu suất của hệ thống

Câu 4: Trình bày về hoạt động của các cổng trong vi xử lý AVR.

Gợi ý

- Vị trí, cấu tạo
- Các thanh ghi
- Các chế độ hoạt động

Đề số 6

Câu 1: Tại sao ta sử dụng phương pháp phân lớp để biểu diễn kiến trúc hệ thống nhúng?

Gợi ý

- Vị trí, cấu tạo
- Các thanh ghi
- Các chế độ hoạt động

Câu 2: Trình bày những đặc điểm phần cứng của một hệ thống nhúng.

Gợi ý

- Trình bày sự khác nhau giữa hệ thống thường và hệ thống nhúng
- Các hạn chế về bộ nhớ
- Hạn chế về kích thước
- Hạn chế về năng lượng
- ….

Câu 3: Thế nào là MiddleWare software, Application software? Cho ví dụ.

Gợi ý

- Trình bày vị trí, chức năng của MiddleWare
- Trình bày vị trí, chức năng của Application
- So sánh hai kiểu phần mềm này

Câu 4: Nêu sự thực thi của các lệnh trong vi xử lý ARM.

Gợi ý

- Trình bày chỉ lệnh xử lý dữ liệu
- Trình bày chỉ lệnh truyền dữ liệu
- Trình bày chỉ lệnh điều khiển dòng