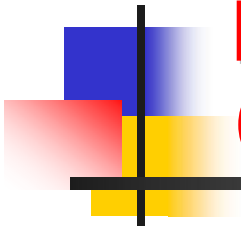


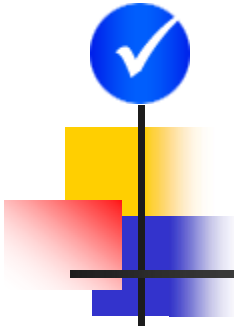
Kiến trúc máy tính
(Computer Architecture)

Chương 2
Kiến trúc phần mềm bộ xử lý

Mục đích, yêu cầu

- **Mục đích:** Giới thiệu các thành phần cơ bản của một hệ thống máy tính, các thành phần của kiến trúc máy tính và kiến trúc phần mềm của máy tính (dạng lệnh, các kiểu định vị và tập lệnh).
- **Yêu cầu:** Sinh viên hiểu về cấu tạo cơ bản của một hệ thống máy tính. Nắm vững ý nghĩa của các thành phân trong kiến trúc phần mềm của máy tính điện tử.

  Phân biệt được hai loại kiến trúc phần mềm:
  - CISC (Complex Instruction Set Computer),
  - RISC (Reduced Instruction Set Computer).

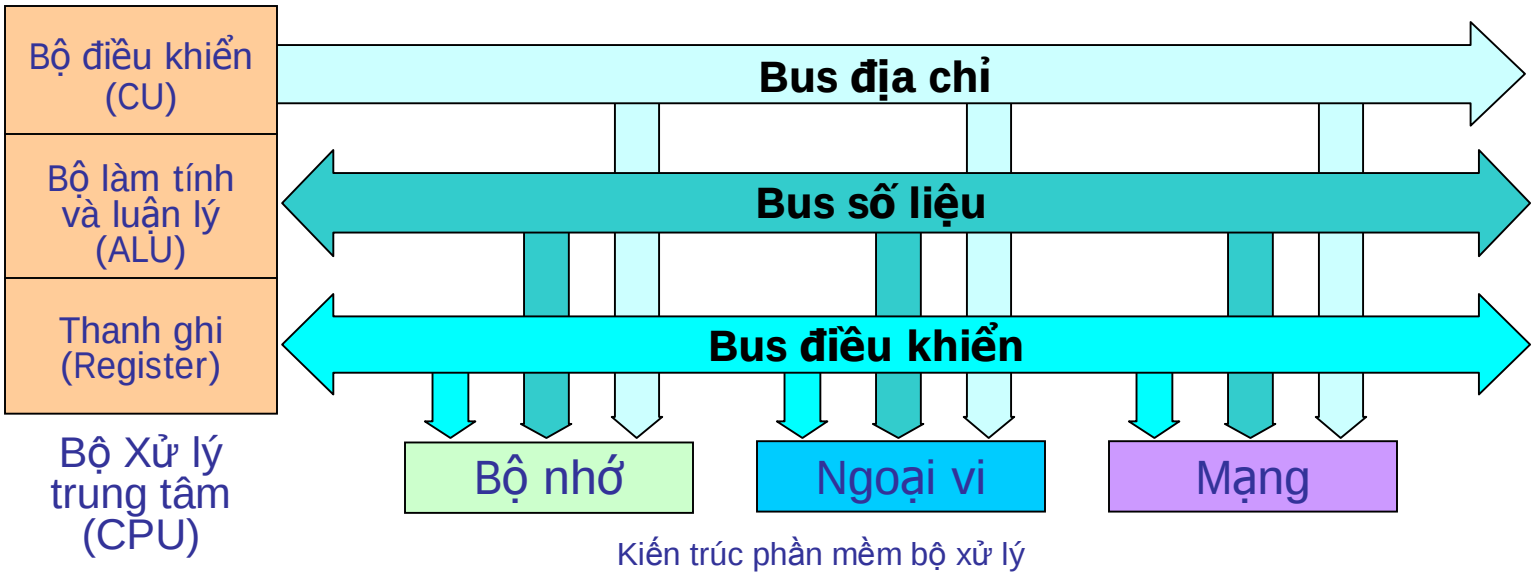


Nội dung

1. Thành phần cơ bản của một máy tính
2. Định nghĩa kiến trúc máy tính
3. Các kiểu thi hành một lệnh
4. Kiểu kiến trúc thanh ghi đa dụng
5. Các kiểu định vị
6. Loại và chiều dài toán hạng
7. Tác vụ mà lệnh thực hiện
8. Kiến trúc RISC
9. Các kiểu định vị trong các bộ xử lý RISC
10. Tập lệnh
11. Ngôn ngữ cấp cao và ngôn ngữ máy

1. Thành phần cơ bản của một máy tính

- Máy tính xử lý thông tin trong bộ xử lý trung tâm (CPU: Central Processing Unit) theo một chương trình có sẵn trong bộ nhớ trong.
- Chương trình gồm có lệnh và số liệu.
  - Lệnh và số liệu do một bộ phận nhập thông tin cung cấp (bàn phím, đĩa).
  - Lệnh được bộ xử lý trung tâm giải mã để biết cách xử lý số liệu.
  - Chương trình cho kết quả ra một bộ phận xuất thông tin (máy in).

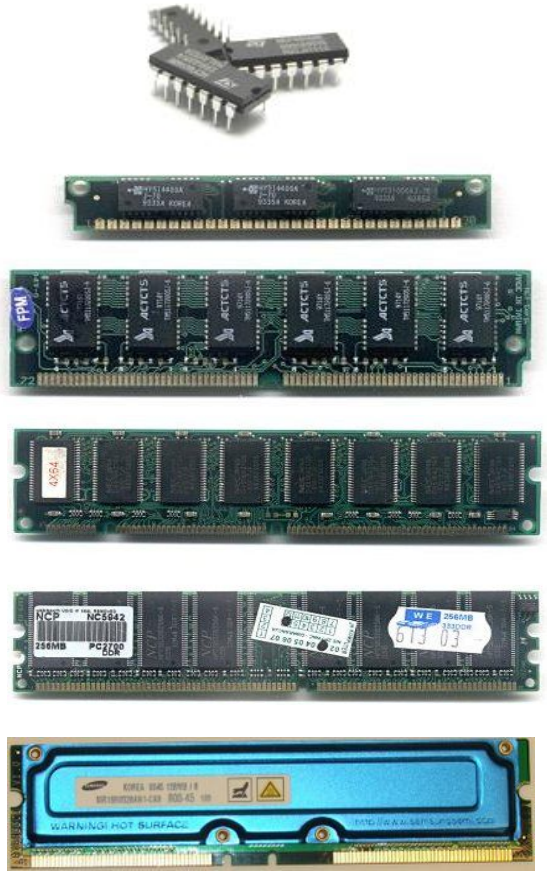


1. Thành phần cơ bản của một máy tính

- **Bộ xử lý trung tâm (CPU):** Đây là bộ phận thi hành lệnh.
  - CPU lấy lệnh từ bộ nhớ trong và lấy các số liệu mà lệnh đó xử lý.
  - Bộ xử lý trung tâm gồm:
    - Phần thi hành lệnh: Gồm bộ làm toán số học & luận lý và các thanh ghi.
    - Phần điều khiển: Thi hành tuần tự các lệnh và tác động vào các mạch điện để thi hành các lệnh

1. Thành phần cơ bản của một máy tính

- **Bộ nhớ trong:** Tập hợp các ô nhớ, mỗi ô nhớ gồm một số bít nhất định và chứa một thông tin được mã hóa thành số nhị phân (Lệnh / số liệu).
  - Mỗi ô nhớ của bộ nhớ trong đều có một địa chỉ.
  - Thời gian thâm nhập bất kỳ ô nhớ nào trong bộ nhớ là như nhau (RAM: Random Access Memory).
  - Độ dài một từ máy tính (8/16/32/64 bits). Mỗi ô nhớ có thể là một byte.

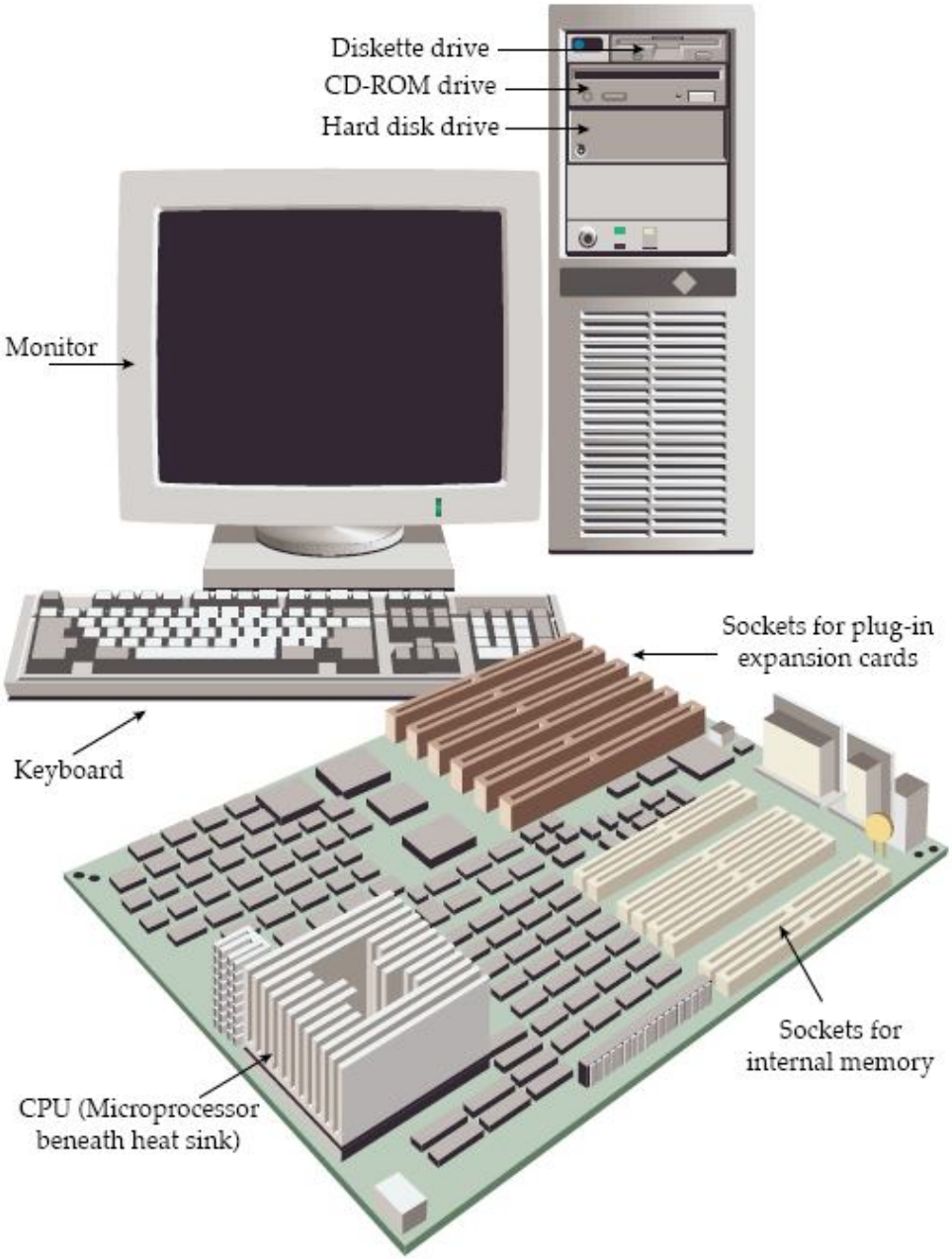


1. Thành phần cơ bản của một máy tính

- **Bộ phận vào ra:** Giao tiếp giữa máy tính và người sử dụng.
  Các bộ phận vào-ra thường gặp:
  - Màn hình (Monitor), bàn phím (Keyboard), con chuột (mouse), máy in (Printer), máy quét (Scanner), … là những bộ phận giúp con người sử dụng máy tính dễ dàng.
  - Bộ nhớ ngoài (Đĩa từ, CDRom, ...) là các bộ phận lưu trữ.
  - Giao diện mạng (**NIC**: **N**etwork **I**nterface **C**ontroller).

1. Thành phần cơ bản của một máy tính

- Một hệ thống máy tính tiêu biểu

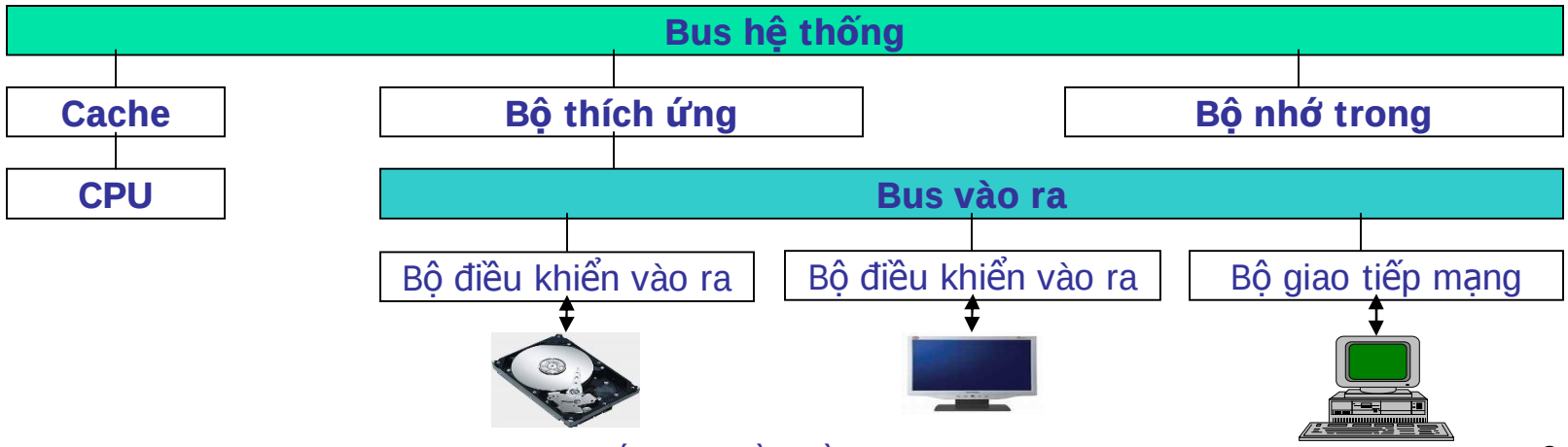


1. Thành phần cơ bản của một máy tính

- **Hệ thống BUS:** Các bộ phận của máy tính được nối với nhau bằng một hệ thống các đường dây liên lạc mà ta gọi là bus.
  - **BUS địa chỉ**: Xác định địa chỉ của các bộ phận.
  - **BUS số liệu:** Trao đổi dữ liệu giữa các bộ phận.
  - **BUS điều khiển:** Điều khiển trao đổi thông tin giữa các bộ phận.

Thông thường người ta phân biệt:

- **Bus hệ thống:** Dùng trao đổi thông tin giữa CPU và bộ nhớ trong
- **Bus vào-ra:** Trao đổi giữa các bộ phận vào-ra và bộ nhớ trong



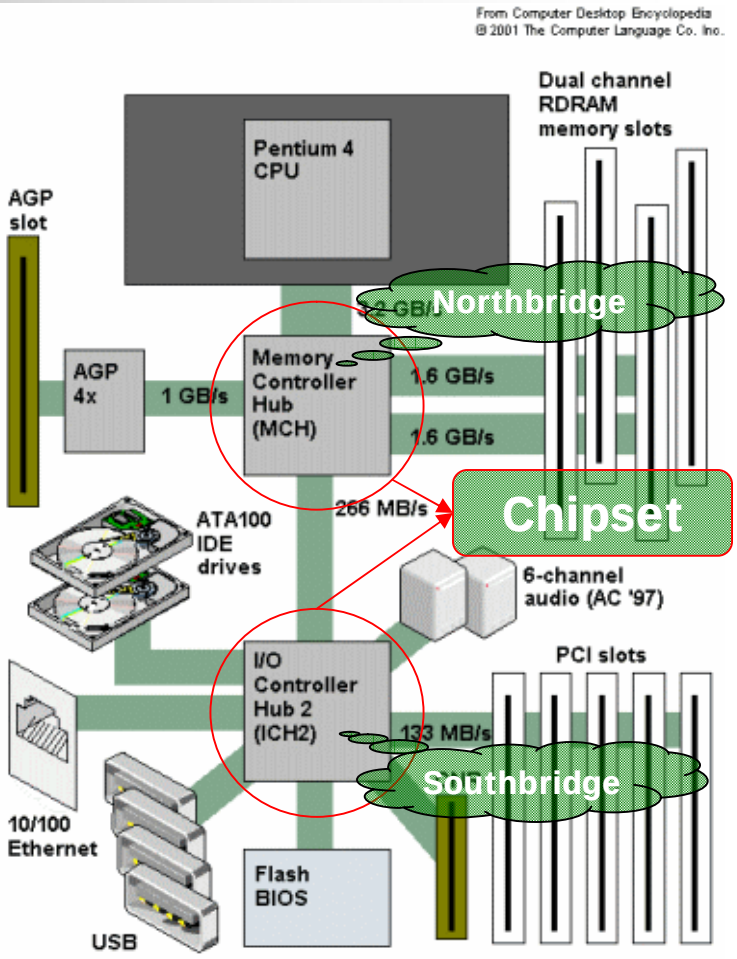
1. Thành phần cơ bản của một máy tính

Ví dụ: Sơ đồ khối máy vi tính.

Minh hoạ hoạt động của Máy tính điện tử

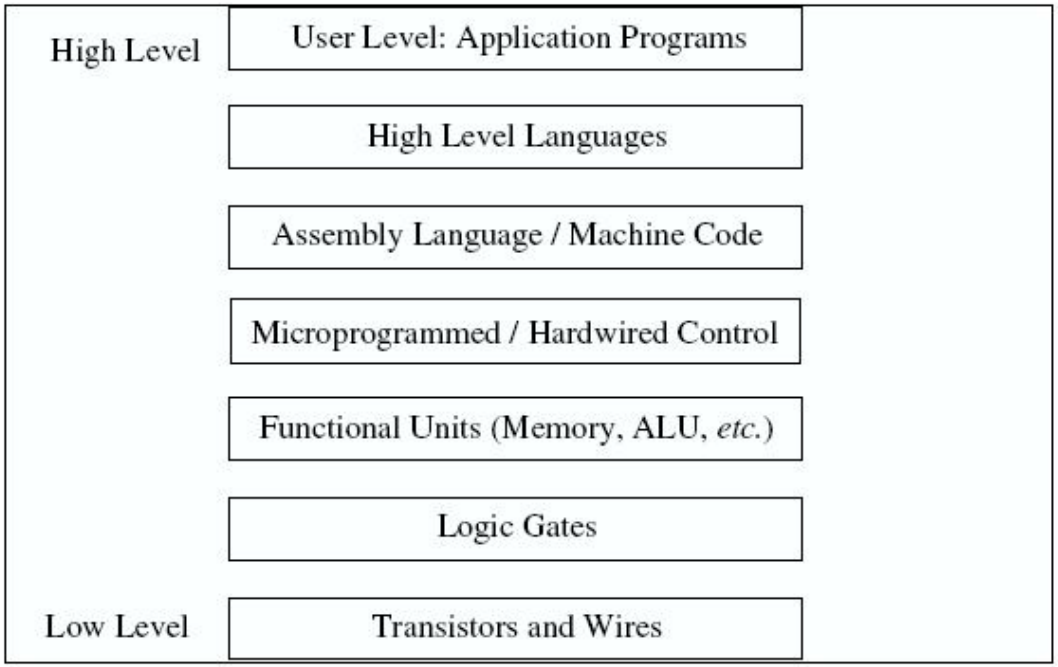
AGP: Accelerated Graphics Port

PCI:Peripheral Compoment Interconnect



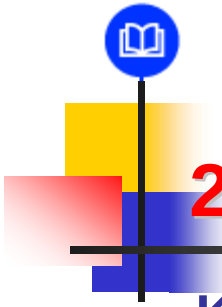
1. Thành phần cơ bản của một máy tính

- **Các cấp của máy tính**



1. Thành phần cơ bản của một máy tính

- **Các cấp của máy tính**
  - Mức chương trình ứng dụng hay người dùng (thân thiện nhất với người dùng, người dùng tương tác với máy tính thông qua việc chạy chương trình, không nhìn thấy bên trong hay các mức thấp hơn)
  - Mức ngôn ngữ cấp cao
  - Mức ngôn ngữ máy hay ngôn ngữ Assembly
  - Mức điều khiển
  - Mức đơn vị chức năng
  - Mức cổng logic, transistor và mạch điện



2. Định nghĩa kiến trúc máy tính

Kiến trúc máy tính bao gồm ba phần:

1. **Kiến trúc phần mềm** của máy tính chủ yếu là kiến trúc phần mềm của bộ xử lý. Nó gồm có:
   - Dạng các lệnh: Loại tác vụ, số toán hạng và chiều dài lệnh.
   - Các kiểu định vị: Cách thức thâm nhập các toán hạng.
   - Tập lệnh: Tập hợp các lệnh mã máy của bộ xử lý.

   Nếu ta thêm vào các thanh ghi của bộ xử lý, thì kiến trúc phần mềm là phần của máy tính mà người lập trình bằng hợp ngữ có thể thấy được.
2. **Phần tổ chức** của máy tính liên quan đến cấu trúc bên trong của bộ xử lý, cấu trúc các bus, các cấp bộ nhớ và các mặt kỹ thuật khác của máy tính.
3. **Lắp đặt phần cứng** của máy tính ám chỉ việc lắp ráp một máy tính dùng các linh kiện điện tử và các bộ phận phần cứng cần thiết.

3. Các kiểu thi hành của một lệnh (1)

- Một lệnh mã máy bao gồm một mã tác vụ và các toán hạng.
- **Chọn số toán hạng cho một lệnh mã máy là một vấn đề then chốt** vì phải có sự cân đối giữa tốc độ tính toán và số các mạch tính toán phải dùng.

  Ví dụ: Lệnh Y := A + B + C + D
  - Có thể được hiện bằng một lệnh mã máy nếu có 3 mạch cộng,
  - Hoặc thực hiện bằng 3 lệnh mã máy nếu chỉ có một mạch cộng.

  Tần số sử dụng một lệnh như trên ít nên không cần phải tốn chi phí lắp đặt 3 mạch cộng. Thường số toán hạng thay đổi từ 0 tới 3.
- **Vị trí của toán hạng cũng được xem xét**. Ba kiểu cơ bản của vị trí các toán hạng đối với những lệnh tính toán trong ALU.
  - Ngăn xếp
  - Thanh ghi tích lũy
  - Thanh ghi đa dụng

  Một vài nhà sản xuất máy tính tuân thủ các kiểu chọn vị trí toán hạng nêu trên, nhưng phần nhiều các bộ xử lý dùng kiểu hỗn tạp.

3. Các kiểu thi hành của một lệnh (2)

C := A + B		
Ngăn xếp	**Thanh ghi tích luỹ**	**Thanh ghi tích đa dụng**
PUSH A PUSH B **ADD** POP C	LOAD A **ADD B** STORE C	LOAD R1,A **ADD B,R1** STORE R1,C

Loại kiến trúc	Lợi điểm	Bất lợi
Ngăn xếp (Stack)	▪ Lệnh ngắn, Ít mã máy ▪ Làm tối thiểu trạng thái bên trong ▪ Dễ dàng tạo ra một bộ biên dịch đơn giản cho kiến trúc ngăn xếp	▪ Thâm nhập không ngẫu nhiên ▪ Mã lệnh không hiệu quả ▪ Khó dùng trong song song & ống dẫn ▪ Khó tạo bộ biên dịch tối ưu
Thanh ghi tích luỹ (Accumulator Register)	▪ Lệnh ngắn ▪ Làm tối thiểu trạng thái bên trong máy tính ▪ Thiết kế dễ dàng	▪ Lưu trong thanh ghi tích luỹ là tạm thời. ▪ Nghẽn ở thanh ghi tích luỹ ▪ Khó dùng trong song song & ống dẫn ▪ Trao đổi nhiều với bộ nhớ
Thanh ghi đa dụng (General Register)	▪ Xử lý nhanh, định vị đơn giản. ▪ Ít thâm nhập bộ nhớ. ▪ Kiểu tổng quát để tạo mã hữu hiệu	▪ Chương trình dài ▪ Số lượng thanh ghi bị giới hạn

4. Kiểu kiến trúc thanh ghi đa dụng

Đối với một lệnh tính toán hoặc logic điển hình (lệnh ALU), có 2 điểm cần nêu lên.

1. **Một lệnh ALU phải có 2 hoặc 3 toán hạng.**
   - Nếu có 3 toán hạng thì một trong các toán hạng chứa kết quả phép tính trên hai toán hạng kia.
   - Nếu có 2 toán hạng thì một trong hai toán hạng phải vừa là toán hạng nguồn, vừa là toán hạng đích.
2. **Phải biết có bao nhiêu toán hạng bộ nhớ.**
   Số toán hạng bộ nhớ có thể thay đổi từ 0 tới 3. Trong nhiều cách tổ hợp có thể có của một lệnh ALU, các máy tính hiện nay chọn một trong 3 kiểu sau :
   - Thanh ghi-thanh ghi (kiểu này còn được gọi nạp - lưu trữ),
   - Thanh ghi - bộ nhớ
   - Bộ nhớ - bộ nhớ.

Kiểu thanh ghi - thanh ghi được nhiều nhà chế tạo máy tính lưu ý vì nó đơn giản và hữu hiệu.

5. Các kiểu định vị

Kiểu định vị định nghĩa cách thức thâm nhập các toán hạng.

- Một vài kiểu xác định cách thâm nhập toán hạng bộ nhớ, nghĩa là cách tính địa chỉ của toán hạng,
- Các kiểu khác xác định các toán hạng nằm trong các thanh ghi.

Chú ý: - Trong các kiểu định vị, khi chuyển đổi dữ liệu nhị phân giữa hai kiểu định địa chỉ liên quan đến ô nhớ, vì mỗi từ máy tính gồm bốn byte, mỗi ô nhớ chứa một byte.

- Như vậy, một từ máy tính được lưu trong bốn ô nhớ liên tiếp trong bộ nhớ trong, có nhiều cách xác một từ máy tính, trong đó, hai cách tiêu biểu nhất là:

Byte thấp được đặt trong ô nhớ có địa chỉ cao (IBM, Motorolla, Sun, HP).

Byte thấp được đặt trong ô nhớ có địa chỉ thấp (Intel, DEC)

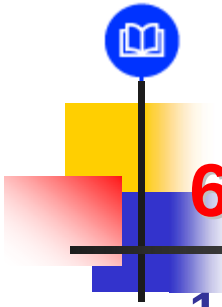
Word address is x for both big-endian and little-endian formats.

Địa chỉ từ là x cho cả hai minh hoạ

Hình II.3: Minh hoạ hai cách sắp xếp địa chỉ trong bộ nhớ

5. Các kiểu định vị

Kiểu định vị	Ví dụ	Giải thích
Thanh ghi	ADD R3,R4	R3 ← R3 + R4
Tức thì	ADD R4,#3	R4 ← R4 + 3
Trực tiếp	ADD R1,(1001)	R1 ← R1 + M[1001]
Gián tiếp (thanh ghi)	ADD R4,(R1)	R4 ← R4 + M[R1]
Gián tiếp (bộ nhớ)	ADD R1,@(R3)	R1 ← R1 + M[M[R3]]
Gián tiếp (thanh ghi + độ dời)	ADD R4,100(R1)	R4 ← R4 + M[R1+100]
Gián tiếp (thanh ghi + thanh ghi)	ADD R3,(R1+R2)	R3 ← R3 + M[R1+R2]
Gián tiếp (t/g nền + t/g chỉ số + độ dời)	ADD R4,100(R2) [R3]	R4 ← R4 + M[100+R2+d*R3]
Tự tăng	ADD R1,(R2)+	R1 ← R1 + M[R2] R2 ← R2 + d
Tự giảm	ADD R1,-(R2)	R2 ← R2 - d R1 ← R1 + M[R2]



6. Loại và chiều dài toán hạng

1. **Loại của toán hạng:** Thường được đưa vào trong mã tác vụ của lệnh. Tuy nhiên một số ít máy tính dùng các nhản để xác định loại toán hạng.
2. **Chiều dài toán hạng:** Thông thường loại của toán hạng xác định luôn chiều dài của nó.

   Toán hạng thường có chiều dài là:

   - Byte (8 bits)
   - Nữa từ máy tính (16 bits)
   - Từ máy tính (32 bits)
   - Từ đôi máy tính (64 bits)

   Đặc biệt, kiến trúc PA của hảng Hewlet Packard có khả năng tính toán với các số thập phân BCD. Một vài bộ xử lý có thể xử lý các chuỗi ký tự.

7. Tác vụ mà lệnh thực hiện

STT	Tác vụ	Diễn giải
1	Tính toán số học và luận lý	Phép tính số nguyên và phép tính luận lý
2	Di chuyển số liệu	Nạp số liệu, lưu giữ số liệu
3	Chuyển điều khiển	Lệnh nhảy, lệnh vòng lặp, gọi ch. trình con…
4	Hệ thống	Gọi hệ điều hành, quản lý bộ nhớ ảo
5	Tính số có dấu chấm động	Các phép tính trên số có dấu chấm động
6	Tính số thập phân	Các phép tính trên số thập phân
7	Tính toán trên chuỗi ký tự	Chuyển, so sánh, tìm kiếm chuỗi ký tự

- Tất cả các loại máy tính đều có 3 loại tác vụ đầu tiên
- Tuỳ theo từng loại máy tính mà nó có từ 0 đến 4 tác vụ còn lại

8. Kiến trúc RISC (Reduced Instruction Set Computer) (1)

- **Kiến trúc CISC:** Kiến trúc với tập lệnh phức tạp CISC (Complex Instruction Set Computer) được nghĩ ra từ những năm 1960.
  - Vào thời ký này, người ta nhận thấy:
    - Các chương trình dịch khó dùng các thanh ghi.
    - Các vi lệnh được thực hiện nhanh hơn các lệnh.
    - Cần thiết phải làm giảm độ dài các chương trình.
  - Nhận xét về kiến trúc CISC:
    - Ưu tiên chọn kiểu định vị: ô nhớ - ô nhớ và ô nhớ - thanh ghi.
    - Câu lệnh phức tạp và dùng nhiều kiểu định vị.
    - Các lệnh có chiều dài thay đổi.
    - Dùng bộ điều khiển vi chương trình.
  - Tiến bộ trong lãnh vực mạch kết (IC) và kỹ thuật dịch chương trình làm cho các nhận định trước đây phải được xem xét lại:
    - Chương trình dịch đã biết sử dụng các thanh ghi và
    - Không khác biệt đáng kể khi sử dụng ô nhớ cho các vi chương trình & chương trình.

8. Kiến trúc RISC (Reduced Instruction Set Computer) (2)

Bộ xử lý	IBM 370/168	DEC 11/780	iAPX 432
Năm sản xuất	1973	1978	1982
Số lệnh	208	303	222
Bộ nhớ vi chương trình	420 KB	480 KB	64 KB
Chiều dài lệnh (bit)	16-48	16-456	6-321
Kỹ thuật chế tạo	ECL-MSI	TTI-MSI	NMOS-VLSI
Cách thực hiện lệnh	Th/g - Th/g Th/g - Bộ nhớ Bộ nhớ - Bộ nhớ	Th/g - Th/g Th/g - Bộ nhớ Bộ nhớ - Bộ nhớ	Ngăn xếp Bộ nhớ - Bộ nhớ
Dung lượng Cache	64 KB	64 KB	0

Đặc tính của một vài máy CISC

8. Kiến trúc RISC (Reduced Instruction Set Computer) (3)

- **Kiến trúc RISC:** Khái niệm về một máy tính với tập lệnh rút gọn RISC vào đầu những năm 1980.

Các máy RISC có các đặc điểm chủ yếu sau:

- Dựa trên tập lệnh cho phép thực hiện kỹ thuật ống dẫn bằng cách thiết kế các lệnh có chiều dài cố định, dạng đơn giản, dễ giải mã.
- Dùng kiểu thực hiện lệnh thanh ghi - thanh ghi. Chỉ có các lệnh ghi hoặc đọc ô nhớ mới cho phép thâm nhập vào ô nhớ.

Định nghĩa mạch xử lý RISC bởi các tính chất sau:

- Có một số ít lệnh (thường thường dưới 100 lệnh).
- Có một số ít kiểu định vị.
- Có một số ít dạng lệnh (một hoặc hai).
- Các lệnh đều có cùng chiều dài.
- Ưu tiên định vị thanh ghi, chỉ có các lệnh ghi hoặc đọc ô nhớ mới thâm nhập vào bộ nhớ.
- Dùng bộ tạo tín hiệu điều khiển bằng mạch điện.
- Bộ xử lý RISC có nhiều thanh ghi để giảm bớt thâm nhập bộ nhớ.
- Bộ xử lý RISC đầu tiên thực hiện các lệnh trong một chu kỳ máy.

8. Kiến trúc RISC (Reduced Instruction Set Computer) (4)

Bộ xử lý	IBM 801	RISC1	MIPS
Năm sản xuất	1980	1982	1983
Số lệnh	120	39	55
Bộ nhớ vi chương trình	0	0	0
Chiều dài lệnh (bit)	32	32	32
Kỹ thuật chế tạo	ECL-MSI	NMOS-VLSI	NMOS-VLSI
Cách thực hiện lệnh	Th/g - Th/g	Th/g - Th/g	Th/g - Th/g

Đặc tính của một vài máy RISC đầu tiên

8. Kiến trúc RISC (Reduced Instruction Set Computer) (5)

Bộ xử lý RISC có các lợi điểm sau:

- Diện tích của bộ xử lý dùng cho bộ điều khiển giảm: Từ 60% (cho các bộ xử lý CISC) xuống còn 10% (cho các bộ xử lý RISC).

⇨Có thể đưa thêm các thanh ghi, các cổng vào ra và bộ nhớ cache.

- Hệ số đều đặn, được định nghĩa là tỉ số giữa tổng số các mạch và số các mạch cần phải vẽ, được tăng lên. Hệ số này là 12 cho MC 68000, 8 cho iAPX 432 nhưng là 25 cho RISC1.

⇨Thời gian để thiết kế bộ điều khiển ít ⇨ Giảm chi phí thiết kế.

- Việc thực hiện một bộ xử lý RISC đơn thể (Module) là có thể được.
- Tốc độ tính toán cao:
  - Nhờ vào việc giải mã lệnh đơn giản,
  - Nhờ có nhiều thanh ghi (ít thâm nhập bộ nhớ),
  - Nhờ thực hiện kỹ thuật ống dẫn liên tục và có hiệu quả (các lệnh đều có thời gian thực hiện giống nhau và có cùng dạng).
- Bộ điều khiển trở nên đơn giản và gọn làm cho ít mắc phải sai sót mà ta gặp thường trong bộ điều khiển.

8. Kiến trúc RISC (Reduced Instruction Set Computer) (6)

Kiến trúc RISC có một số bất lợi:

- Chương trình dài so với bộ xử lý CISC, do các nguyên nhân:
  - Cấm thâm nhập bộ nhớ đối với tất cả các lệnh (trừ lệnh đọc và ghi vào bộ nhớ) ⇨ Phải dùng nhiều lệnh để làm một công việc nhất định; Phải tính địa chỉ hiệu dụng vì ít cách định vị.
  - Tập lệnh có ít lệnh nên các lệnh không có sẵn phải được thay thế bằng một chuỗi lệnh của bộ xử lý RISC.
  - Các chương trình dịch gặp nhiều khó khăn vì có ít lệnh làm cho có ít lựa chọn để dịch các cấu trúc của chương trình gốc.
  - Sự cứng nhắc của kỹ thuật ống dẫn cũng gây khó khăn.
  - Có ít lệnh trợ giúp cho ngôn ngữ cấp cao (Bộ xử lý CISC trợ giúp mạnh hơn các ngôn ngữ cao cấp nhờ tập lệnh phức tạp)
- Các tiến bộ gần đây cho phép xếp đặt trong một vi mạch, một bộ xử lý RISC nền và nhiều toán tử chuyên dùng (CPU Intel 860: một bộ xử lý RISC, bộ làm tính số lẻ, và bộ tạo tín hiệu đồ họa).

9. Các kiểu định vị trong các bộ xử lý RISC (1)

- **Kiểu định vị thanh ghi:** Đây là kiểu định vị thường dùng cho các bộ xử lý RISC, các toán hạng nguồn và kết quả trong thanh ghi mà số thứ tự được nêu ra trong lệnh.

MIPS	**Op code 6**						**Nguồn 1 5**					**Nguồn 2 5**					**Đích 5**					**Dịch chuyển 5**					**Hàm 6**			
SPARC	**Op. 2**		**Đích 5**					**Op code 6**					**Nguồn 1 5**				**0 1**	**Khoảng trống 8**								**Nguồn 2 5**			
POWER PC	**Op code 6**						**Đích 5**					**Nguồn 1 5**					**Nguồn 2 5**					**Op code mở rộng 10**								**0 1**
ALPHA	**Op code 6**						**Nguồn 1 5**					**Nguồn 2 5**					**3**			**0 1**	**Op. mở rộng 7**							**Đích 5**		

Dạng lệnh trong kiểu định vị thanh ghi – thanh ghi của vài CPU RISC

Ví dụ: *Lệnh ADD R4, R15, R9 của CPU Power PC có mã tác vụ là 011010*

0	1	1	0	1	0	0	0	1	0	0	0	1	1	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
Op code (6bit)						Đích (5bit)					Nguồn 1 (5bit)					Nguồn 2 (5bit)					Op code mở rộng (10bit)										1

9. Các kiểu định vị trong các bộ xử lý RISC (2)

- **Kiểu định vị tức thì:** Toán hạng là một số có dấu, được chứa ngay trong lệnh.

MIPS	**Op code 6**		**Nguồn 1 5**		**Đích 5**		**Số có dấu (toán hạng tức thì) 16**					
SPARC	**Op. 2**	**Đích 5**		**Op code 6**		**Nguồn 1 5**	**1 1**	**Số có dấu (toán hạng tức thì) 13**				
POWER PC	**Op code 6**		**Đích 5**		**Nguồn 1 5**		**Số có dấu (toán hạng tức thì) 16**					
ALPHA	**Op code 6**		**Nguồn 1 5**		**Số nguyên dương 8**		**1 1**	**Op. mở rộng 7**			**Đích 5**	

Dạng lệnh trong kiểu định vị thanh ghi – tức thì của vài CPU RISC

Ví dụ: *Lệnh ADD R4,R15,#-5620 của CPU PowerPC, mã tác vụ là 011010*

0	1	1	0	1	0	0	0	1	0	0	0	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	0	1	1	0	0
Op code (6bit)						Đích (5bit)					Nguồn (5bit)					Số có dấu (toán hạng tức thì 16bit)															

9. Các kiểu định vị trong các bộ xử lý RISC (3)

- **Kiểu định vị trực tiếp:** Địa chỉ toán hạng nằm ngay trong lệnh. Kiểu định vị trực tiếp được dùng cho các biến của hệ điều hành.
- **Kiểu định vị gián tiếp bằng thanh ghi + độ dời:** Địa chỉ toán hạng được tính: Địa chỉ toán hạng = Thanh ghi + độ dời.

  Kiểu định vị trực tiếp là một trường hợp đặc biệt của kiểu này khi thanh ghi (địa chỉ) = 0 (R0 hoặc R31 được mắc vào điện thế thấp)

MIPS	**Op code 6**						**Th/g D.C 5**					**Th/g S.L 5**					**Độ dời có dấu 16**															
SPARC	**Op. 2**		**Th/g S.L 5**					**Op code 6**					**Th/g D.C 5**					**1 1**	**Độ dời có dấu 13**													
POWER PC	**Op code 6**						**Th/g S.L 5**					**Th/g D.C 5**					**Độ dời có dấu 16**															
ALPHA	**Op code 6**						**Th/g S.L 5**					**Th/g D.C 5**					**Độ dời có dấu 16**															
Dạng lệnh thâm nhập bộ nhớ trong của vài CPU RISC																																

Ví dụ: *Lệnh Store R4,100(R7) của CPU PowerPC, mã tác vụ là 001010*

0	0	1	0	1	0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
Op code (6bit)						Th/g S.L (5bit)					Th/g D.C (5bit)					Độ dời có dấu (16bit)															

10. Tập lệnh (1)

- Tập lệnh là tập hợp các lệnh mã máy của bộ xử lý.
- Mục tiêu của phần này là dùng các ví dụ trích từ các kiến trúc phần mềm được dùng nhiều nhất, để cho thấy các kỹ thuật ở mức ngôn ngữ máy dùng để thi hành các cấu trúc trong ngôn ngữ cấp cao.
- Để minh họa ta dùng cú pháp lệnh trong hợp ngữ sau đây :

  <Từ gợi nhớ mã lệnh> <Thanh ghi đích>,<Thanh ghi nguồn 1>,<Thanh ghi nguồn 2>

  - Từ gợi nhớ mã lệnh mô tả ngắn gọn tác vụ phải thi hành trên các thanh ghi nguồn, kết quả được lưu giữ trong thanh ghi đích.
  - Mỗi lệnh của ngôn ngữ cấp cao được xây dựng bằng một lệnh mã máy hoặc một chuỗi nhiều lệnh mã máy.
- Lệnh nhảy (GOTO) được thực hiện bằng các lệnh hợp ngữ về nhảy (JUMP) hoặc lệnh hợp ngữ về vòng. Chúng ta phân biệt:
  - Lệnh nhảy làm cho bộ đếm chương trình được nạp vào địa chỉ tuyệt đối phải nhảy đến (PC ← địa chỉ tuyệt đối phải nhảy tới),
  - Lệnh vòng theo đó ta chỉ cần cộng thêm một độ dời vào bộ đếm chương trình (PC ← PC + độ dời).

10. Tập lệnh (2)

- **Gán trị:** Việc gán trị được thực hiện nhờ một số lệnh mã máy (gồm cả gán trị cho biểu thức số học và logic). Cho kiến trúc RISC, ta có:
  - **Lệnh bộ nhớ:**

LOAD	Ri, (địa chỉ)	M[địa chỉ] ➔ Ri
STORE	Ri, (địa chỉ)	Ri ➔ M[địa chỉ]

  - **Lệnh tính toán số nguyên** trên nội dung của hai thanh ghi Ri, Rj và xếp kết quả vào trong Rk:

ADD	(cộng)
ADDD	(cộng số có dấu chấm động, chính xác kép)
SUB	(trừ)
SUBD	(trừ số có dấu chấm động, chính xác kép)
MUL	(nhân)
DIV	(chia)

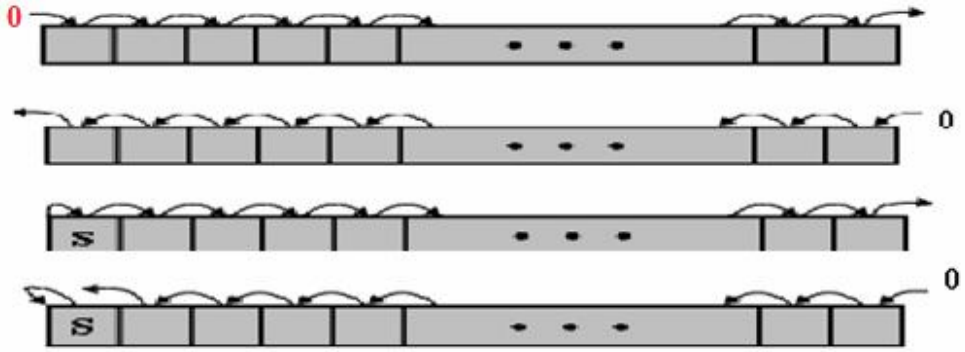
  - **Lệnh logic** thực hiện phép tính logic cho từng bít một:

AND	(lệnh VÀ)
OR	(lệnh HOẶC)
XOR	(lệnh HOẶC LOẠI)
NEG	(lệnh lấy số bù 1)

10. Tập lệnh (3)

Các lệnh dịch chuyển số học hoặc logic (SHIFT):

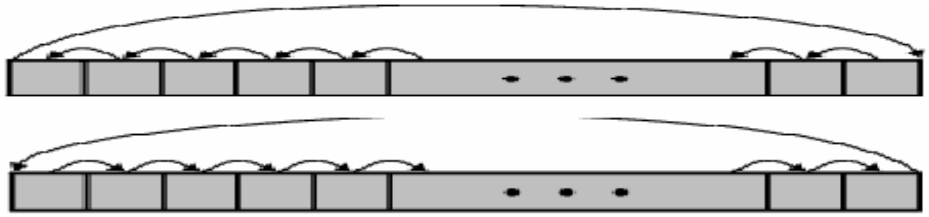
- Dịch phải logic
- Dịch trái logic
- Dịch phải số học
- Dịch trái số học



Quay vòng (ROTATE):

- Có hoặc không có số giữ ở ngã vào,
- Sang phải hoặc sang trái.
- Thực hiện trên một thanh ghi và kết quả lưu trong thanh ghi khác.
- Số lần dịch chuyển thường được xác định trong thanh ghi thứ ba.

- **Quay trái**
- **Quay phải**



10. Tập lệnh (5)

Có hai kỹ thuật cơ bản để ghi nhớ các bít trạng thái

- Cách thứ nhất là ghi các bít trạng thái trong một thanh ghi đa dụng.
  Ví dụ lệnh: CMP Rk, Ri, Rj Thực hiện phép tính Ri-Rj không ghi kết quả phép trừ, ghi các bít trạng thái vào thanh ghi Rk.
  - **Điểm lợi:** Lưu trữ nhiều trạng thái để dùng về sau.
  - **Điểm bất lợi:** Phải dùng một thanh ghi để ghi lại trạng thái.
- Cách thứ hai là để các bít trạng thái vào một thanh ghi đặc biệt.
  Lưu giữ nội dung thanh ghi này được giải quyết bằng nhiều cách:
  - Trong kiến trúc SPARC chỉ có một số giới hạn lệnh được phép thay đổi thanh ghi trạng thái.
    Ví dụ:
    Lệnh ADDCC phép cộng và làm thay đổi thanh ghi trạng thái.
    Lệnh SUBCC phép trừ và làm thay đổi thanh ghi trạng thái.
  - Trong kiến trúc PowerPC thanh ghi trạng thái được phân thành 8 trường 4 bít, vậy là đã phân thành 8 thanh ghi trạng thái con.

10. Tập lệnh (4)

Lệnh có điều kiện:

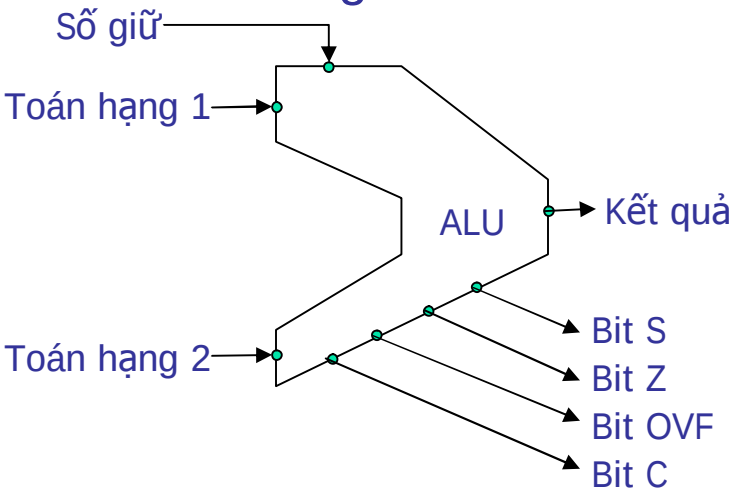
Lệnh có điều kiện có dạng :

Nếu <điều kiện> thì <chuỗi lệnh 1> nếu không <chuỗi lệnh 2>

(IF <condition> THEN <instructions> ELSE <instructions>)

Lệnh này phải ghi nhớ điều kiện và nhảy vòng nếu điều kiện thỏa.

a) Ghi nhớ điều kiện: Bộ làm tính ALU cung cấp kết quả ở ngã ra tùy theo các ngã vào và phép tính cần làm. Nó cũng cho một số thông tin khác về kết quả dưới dạng các bít trạng thái.



Trong các bít trạng thái ta có:

- **Bít dấu S** (1 nếu kết quả âm),
- **Bít trắc nghiệm zero Z** (1 nếu kết quả bằng 0).
- **Bít tràn OVF** (overflow) (1 nếu không đủ khả năng lưu kết quả).
- **Bít số giữ C** (carry) (1 nếu số giữ ở ngã ra là 1)
- …

Các bít trên thường được gọi là bít mã điều kiện.

10. Tập lệnh (6)

b) Nhảy vòng:

- Lệnh nhảy / nhảy vòng có điều kiện (thực hiện khi điều kiện thoả).

  Ví dụ: Giả sử trạng thái sau khi bộ xử lý thi hành một tác vụ, được lưu trử trong thanh ghi, và bộ xử lý thi hành các lệnh sau :

  1. CMP R4, R1, R2 : So sánh R1 và R2 và lưu giữ trạng thái trong R4
  2. BGT R4, +2 : Nhảy bỏ 2 lệnh nếu R1 > R2
  3. ADD R3, R0, R2 : R0 có giá trị 0. Chuyển nội dung của R2 vào R3
  4. BRA +1 : Nhảy bỏ 1 lệnh
  5. ADD R3, R0, R1 : Chuyển nội dung R1 vào R3
  6. Lệnh kế

  - Nếu R1 > R2 thì chuỗi lệnh được thi hành là 1, 2, 5, 6 được thi hành,
  - Nếu không thì chuỗi lệnh 1, 2, 3, 4, 6 được thi hành.
  - Các lệnh thực hiện công việc: Nếu R1>R2 thì R3=R1 nếu không R3=R2

- Lưu ý: Các lệnh nhảy làm giảm tốc độ thi hành lệnh, trong các CPU hiện đại dùng kỹ thuật ống dẫn. Trong một vài bộ xử lý người ta dùng lệnh di chuyển có điều kiện để tránh dùng lệnh nhảy trong một vài trường hợp.

  Ví dụ được viết lại :

  1. CMP R4, R1, R2 : So sánh R1 và R2 và để các bít trạng thái trong R4.
  2. ADD R3, R0, R2 : Di chuyển R2 vào R3
  3. MGT R4, R3, R1 : (MGT: Move if greater than). Nếu R1>R2 ⇨ R1→ R3

10. Tập lệnh (7)

- **Vòng lặp:** Lệnh vòng lặp được thực hiện nhờ lệnh nhảy có điều kiện. Quản lý số lần lặp lại bằng một bộ đếm vòng lặp và kiểm tra bộ đếm sau mỗi vòng lặp để xem đã đủ số vòng cần thực hiện hay chưa?

  Bộ xử lý PowerPC có một lệnh quản lý vòng lặp:

  BNCT Ri, độ dời (Thanh ghi Ri chứa số lần lặp lại).

  Lệnh làm công việc sau: Ri := Ri –1

  Nếu Ri=0, PC:=PC+độ dời.

  Nếu không thì tiếp tục thi hành lệnh lặp.

10. Tập lệnh (8)

- **Thâm nhập bộ nhớ ngăn xếp:**

Ngăn xếp là một tổ chức bộ nhớ sao cho ta chỉ có thể đọc một từ ở đỉnh ngăn xếp hoặc viết một từ vào đỉnh ngăn xếp. Địa chỉ của đỉnh ngăn xếp chứa trong một thanh ghi đặc biệt gọi là con trỏ ngăn xếp SP (stack pointer).

```
- Lệnh PUSH: SP := SP + 1    ; Tăng thanh ghi SP một từ.
             M(SP) := Ri     ; Lưu Ri vào ngăn xếp.
- Lệnh POP:  Ri := M(SP)     ; Đọc ngăn xếp → Ri.
             SP := SP – 1    ; Giảm thanh ghi SP một từ.
```

Ví dụ: Trong các bộ xử lý RISC, thanh ghi R30 là con trỏ ngăn xếp.

Viết vào ngăn xếp:

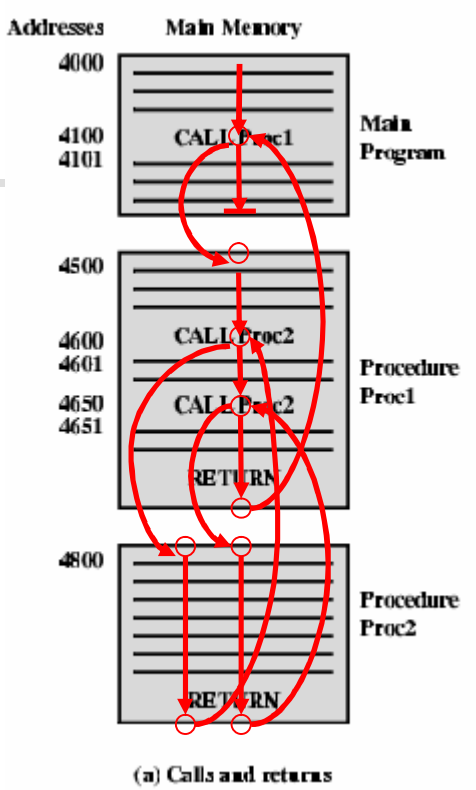
```
ADDI    R30,R30,4    ; Tăng SP lên 4 bytes vì một từ dài 32 bít
STORE   Ri, (R30)    ; Viết Ri  vào đỉnh ngăn xếp
```

Lấy ra khỏi ngăn xếp:

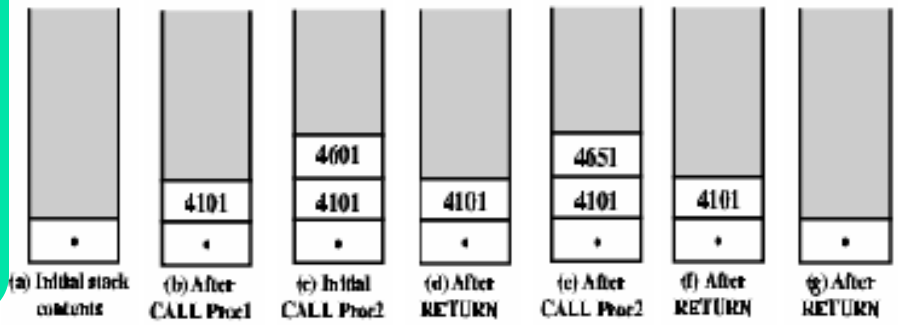
```
LOAD    Ri, (R30)    ; Lấy số liệu ở đỉnh ngăn xếp nạp vào Ri
SUBI    R30, R30,4   ; Giảm SP một từ (4 bytes).
```

10. Tập lệnh (9)

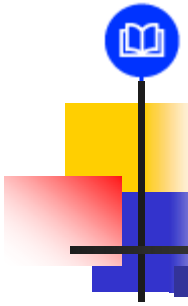
- **Các thủ tục:**
  - Các thủ tục được gọi từ bất cứ nơi nào của chương trình nhờ lệnh gọi thủ tục CALL.
  - Để khi chấm dứt việc thi hành thủ tục, chương trình gọi được tiếp tục bình thường, ta cần lưu địa chỉ trở về. Khi chấm dứt thủ tục, lệnh RET nạp địa chỉ trở về vào PC.
  - Trong kiến trúc CISC, địa chỉ trở về được giữ ở ngăn xếp.



(a) Calls and returns



(a) Initial stack contents (b) After CALL Proc1 (c) Initial CALL Proc2 (d) After RETURN (e) After CALL Proc2 (f) After RETURN (g) After RETURN



10. Tập lệnh (10)

- **Các thủ tục (tt)**
  - Trong các kiến trúc RISC, một thanh ghi đặc biệt (R31) dùng để lưu địa chỉ trở về.
  - Lệnh gọi thủ tục là một lệnh loại JMPL Ri:

    Gồm các thao tác sau:

    ```
    R31 := PC          ; lưu địa chỉ trở về trong R31
    PC := Ri           ; nhảy tới địa chỉ thủ tục (Ri)
    ```

  - Lệnh trở về (chấm dứt thủ tục): JMP R31

    ```
    PC := R31          ; R31: địa chỉ trở về
    ```

10. Tập lệnh (11)

- *Lưu ý:*
  - **Thanh ghi lưu địa chỉ trở về chỉ áp dụng cho thủ tục cuối.**
  - **Các thủ tục có thể gọi thủ tục khác, ta dùng bộ nhớ ngăn xếp**
    - **Gọi thủ tục 1: JMPL Ri**

      R31 := PC ; lưu địa chỉ trở về trong R31

      PC := Ri ; nhảy tới địa chỉ thủ tục (Ri)
    - **Gọi thủ tục 2:**

      ADDI R30, R30,4 ; R30 là con trỏ ngăn xếp.

      STORE R31, (R30) ; Lưu giữ địa chỉ trở về chương trình chính.

      JMPL Rj ; Gọi thủ tục 2

      R31 := PC ; Lưu địa chỉ trở về thủ tục 1

      PC := Rj ; Gọi thủ tục.
    - **Trở về thủ tục 1: JMP R31**

      PC:=R31
    - **Trở về chương trình chính:**

      LOAD R31, (R30) ; Phục hồi địa chỉ trở về

      SUBI R30, R30,4 ; Cập nhật con trỏ ngăn xếp

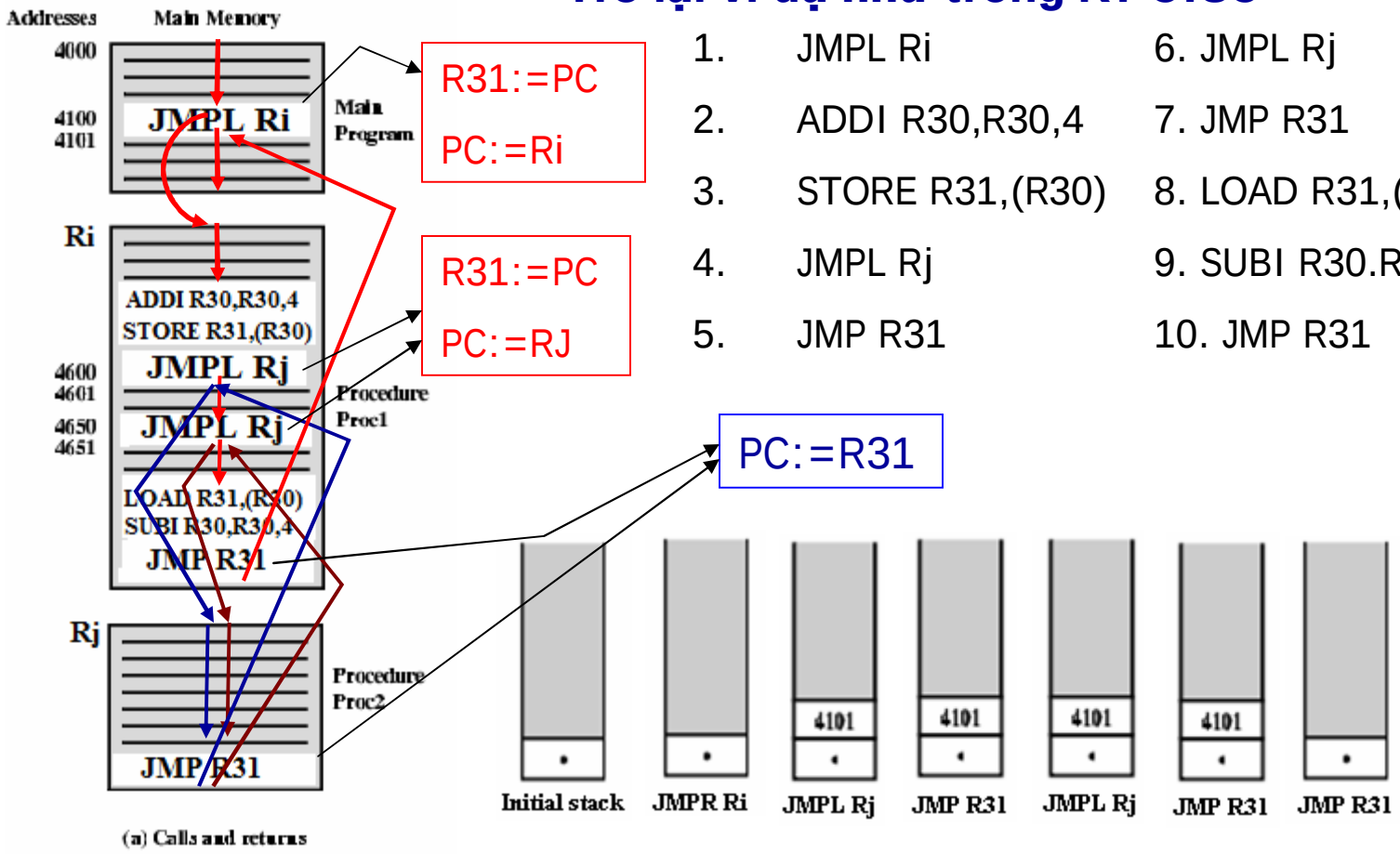
      JMP R31 ; Trở về chương trình gọi

      PC:=R31

      Truyền tham số từ thủ tục gọi đến thủ tục bị gọi có thể thực hiện bằng cách dùng các thanh ghi (ít tham số) hoặc dùng ngăn xếp.

10. Tập lệnh (12)

- **Trở lại ví dụ như trong KT CISC**



1. JMPL Ri
2. ADDI R30,R30,4
3. STORE R31,(R30)
4. JMPL Rj
5. JMP R31
6. JMPL Rj
7. JMP R31
8. LOAD R31,(R30)
9. SUBI R30.R30,4
10. JMP R31

11. Ngôn ngữ cao cấp và ngôn ngữ máy (1)

Giá hệ thống tin học = *Giá máy tính* + *Giá phần mềm hệ thống & Phần mềm ứng dụng.*

- **Ngôn ngữ cấp cao:** Ngôn ngữ cấp cao dùng các lệnh có cấu trúc gần với ngôn ngữ thông thường hơn ngôn ngữ máy.
- Các ngôn ngữ cấp cao nổi tiếng:
  - FORTRAN dùng cho tính toán khoa học,
  - COBOL dùng cho quản lý,
  - LISP và PROLOG dùng trong trí tuệ nhân tạo,
  - PASCAL, C, ADA ...
- Điểm chính của ngôn ngữ cấp cao là sự cô động và sự độc lập.

Sự cô đọng:

- Mỗi lệnh mô tả ngắn gọn, một công việc của bộ xử lý.
- Thể hiện tất cả các công việc thông qua một số hữu hạn các lệnh.

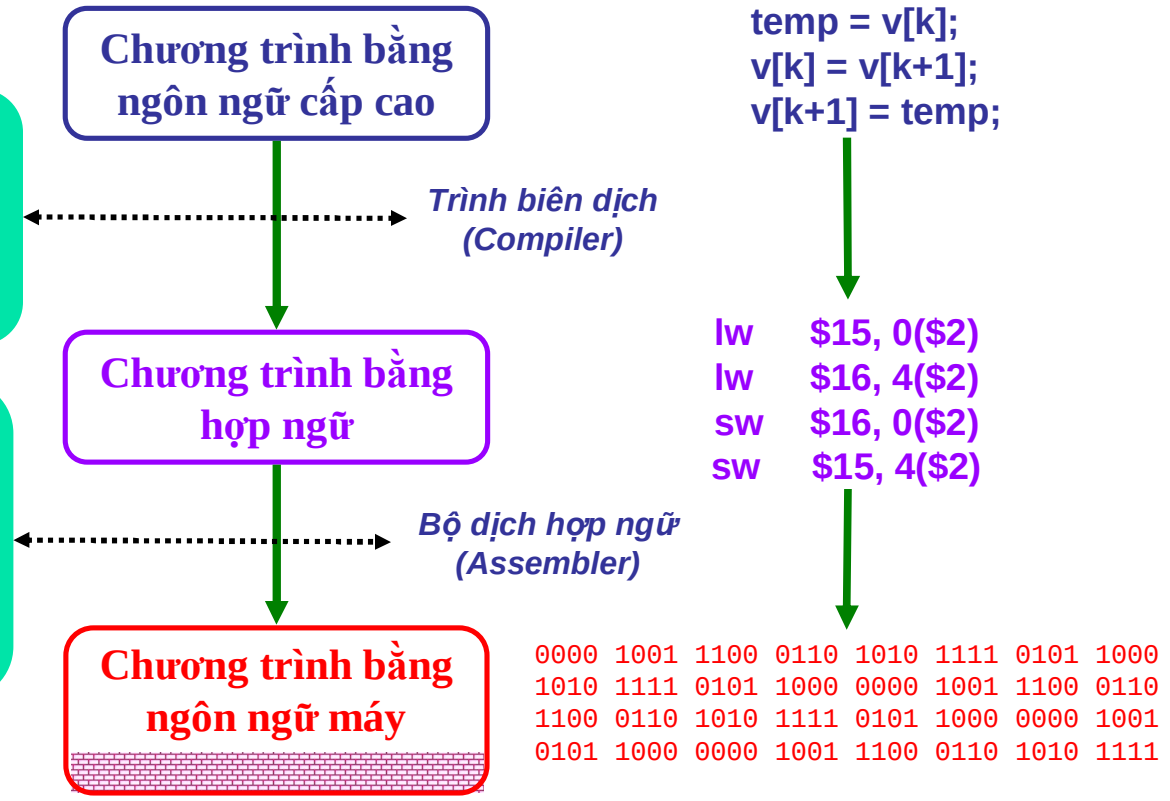
Sự độc lập:

- Có thể được thi hành trên mọi kiến trúc phần mềm của bộ xử lý,
- Phải có chương trình dịch để dịch ngôn ngữ cấp cao ⇨ mã máy.

11. Ngôn ngữ cao cấp và ngôn ngữ máy (2)

Một trình biên dịch chuyển đổi ngôn ngữ cấp cao sang dạng hợp ngữ.

Một bộ dịch hợp ngữ chuyển đổi từ dạng hợp ngữ sang ngôn ngữ máy để máy tính có thể thực hiện.

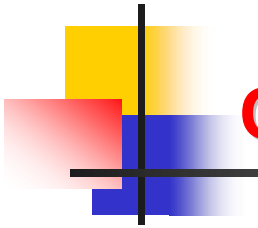


11. Ngôn ngữ cao cấp và ngôn ngữ máy (3)

- **Quan hệ của chương trình dịch đối với ngôn ngữ máy.**
  - Để chương trình ngôn ngữ máy thực hiện một cách hữu hiệu thì chương trình dịch phải dịch hữu hiệu các lệnh của ngôn ngữ cấp cao thành lệnh mã máy.
  - Kiến trúc phần mềm rất quan trọng đối với chương trình dịch.
  - Kiến trúc phần mềm làm giảm nhẹ công việc của chương trình dịch
  - 1980.., Công nghệ viết chương trình dịch, có sự chuyển hướng:

Kiến trúc phần mềm phức tạp ***Bộ xử lý kiểu CISC***	⇨	**Kiến trúc phần mềm đơn giản và hữu hiệu** ***Bộ xử lý kiểu RISC***

- Với những tiến bộ của công nghệ chế tạo máy tính, của công nghệ viết chương trình dịch và của công nghệ lập trình, người ta đang tiến tới chế tạo các kiến trúc phần mềm hấp dẫn hơn trong tương lai.

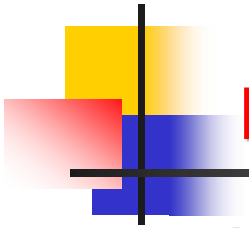


Câu hỏi ?

1. Các thành phần cơ bản của máy tính điện tử.
2. Nhiệm vụ bộ nhớ trong của một máy tính là gì?
3. Nhiệm vụ của các bộ phận vào ra của một máy tính là gì?
4. Nhiệm vụ bộ xử lý trung tâm của một máy tính là gì?
5. Nhiện vụ của hệ thống bus trong máy tính điện tử?
6. Nhiện vụ của bus địa chỉ trong máy tính điện tử?
7. Nhiện vụ của bus dữ liệu trong máy tính điện tử?
8. Nhiện vụ của bus điều khiển trong máy tính điện tử?
9. Kiến trúc phần mềm của máy tính bao gồm các thành phần nào?
10. Tai sao việc chọn số toán hạng cho một lệnh mã máy là vấn đề then chốt?
11. Tại sao ngày nay các nhà sản xuất máy tính có khuynh hướng dùng kiến trúc phần mềm thanh ghi đa dụng?
12. Kiến trúc phần mềm loại ngăn xếp có điểm lợi & bất lợi gì?
13. Kiến trúc phần mềm kiểu thanh ghi tích luỹ có điểm lợi & bất lợi gì?
14. Kiến trúc phần mềm kiểu thanh ghi đa dụng có điểm lợi & bất lợi gì?

Câu hỏi ?

15. Vị trí toán hạng trong các kiểu định vị tức thời, thanh ghi, trực tiếp ?
16. Trong kiểu định vị thanh ghi, gián tiếp thanh ghi, gián tiếp bộ nhớ, thanh ghi chứa nội dung gì ?
17. Đặc điểm của bộ xử lý CISC ?
18. Đặc điểm của bộ xử lý RISC ?
19. Lợi điểm & Nhược điểm của bộ xử lý RISC ?
20. Trong các kiểu định vị của CPU RISC các toán hạng thanh ghi có chiều dài là 5 bit. Nội dung mà nó chứa là gì ?
21. Trong kiểu định vị thanh ghi cộng độ dời của CPU RISC, khi giá trị của thanh ghi bằng 0 thì nó trở thành kiểu định vị gì ?
22. Ý nghĩa của các bit trạng thái (Zero, Sign, Overflow, Carry) ?.
23. Trong các phép so sánh, người ta thường sử dụng các bit trạng thái nào?
24. Lợi điểm và bất lợi của các giải pháp lưu trử các bit trạng thái ?
25. Trong kiến trúc RISC thường dùng một thanh ghi đặc biệt (R31) để lưu giữ địa chỉ trở về. Giải pháp này chỉ được áp dụng cho các thủ tục nào?
26. Tính cô động và tính độc lập của ngôn ngữ cao cấp thể hiện như thế nào?



Bài tập

Viết lệnh các lệnh mã máy cho bộ xử lý Power PC.

1. Load R25, (1000); Cho biết mã tác vụ Load: 001001
2. Load R25, 100(R15); Cho biết mã tác vụ Load: 001001
3. Store R10, (R24); Cho biết mã tác vụ Store: 001010
4. Add R4, R15, R11; Cho biết mã tác vụ Add: 011010
5. Sub R3, R17, # -1500; Cho biết mã tác vụ Sub: 011011

Bài tập (Hướng dẫn)

Load R25, (1000) — **Mã tác vụ lệnh Load: 001001**

d_{31}	d_{30}	d_{29}	d_{28}	d_{27}	d_{26}	d_{25}	d_{24}	d_{23}	d_{22}	d_{21}	d_{20}	d_{19}	d_{18}	d_{17}	d_{16}	d_{15}	d_{14}	d_{13}	d_{12}	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
Opcode						Th/g SL					Th/g DC					Do doi 16 bit														

Load R25, 100(R15) — **Mã tác vụ lệnh Load: 001001**

d_{31}	d_{30}	d_{29}	d_{28}	d_{27}	d_{26}	d_{25}	d_{24}	d_{23}	d_{22}	d_{21}	d_{20}	d_{19}	d_{18}	d_{17}	d_{16}	d_{15}	d_{14}	d_{13}	d_{12}	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
Opcode						Th/g SL					Th/g DC					Do doi 16 bit														

Store R10, (R24) — **Mã tác vụ lệnh Store: 001010**

d_{31}	d_{30}	d_{29}	d_{28}	d_{27}	d_{26}	d_{25}	d_{24}	d_{23}	d_{22}	d_{21}	d_{20}	d_{19}	d_{18}	d_{17}	d_{16}	d_{15}	d_{14}	d_{13}	d_{12}	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
Opcode						Th/g SL					Th/g DC					Do doi 16 bit														

Add R4, R15, R11 — **Mã tác vụ lệnh Add: 011010**

d_{31}	d_{30}	d_{29}	d_{28}	d_{27}	d_{26}	d_{25}	d_{24}	d_{23}	d_{22}	d_{21}	d_{20}	d_{19}	d_{18}	d_{17}	d_{16}	d_{15}	d_{14}	d_{13}	d_{12}	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
Opcode						Th/g Dich					Th/g Nguon 1					Th/g Nguon 2					Opcode mo rong									1

Sub R3, R17, # -1500 — **Mã tác vụ lệnh Sub: 011011**

d_{31}	d_{30}	d_{29}	d_{28}	d_{27}	d_{26}	d_{25}	d_{24}	d_{23}	d_{22}	d_{21}	d_{20}	d_{19}	d_{18}	d_{17}	d_{16}	d_{15}	d_{14}	d_{13}	d_{12}	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
Opcode						Th/g Dich					Th/g Nguon 1					So co dau 16 bit														

Bài tập (Hướng dẫn)

Load R25, (1000) Mã tác vụ lệnh Load: 001001

d_{31}	d_{30}	d_{29}	d_{28}	d_{27}	d_{26}	d_{25}	d_{24}	d_{23}	d_{22}	d_{21}	d_{20}	d_{19}	d_{18}	d_{17}	d_{16}	d_{15}	d_{14}	d_{13}	d_{12}	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
0	0	1	0	0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0
Opcode						Th/g SL					Th/g DC					Do doi 16 bit																

Load R25, 100(R15) Mã tác vụ lệnh Load: 001001

d_{31}	d_{30}	d_{29}	d_{28}	d_{27}	d_{26}	d_{25}	d_{24}	d_{23}	d_{22}	d_{21}	d_{20}	d_{19}	d_{18}	d_{17}	d_{16}	d_{15}	d_{14}	d_{13}	d_{12}	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
0	0	1	0	0	1	1	1	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
Opcode						Th/g SL					Th/g DC					Do doi 16 bit																

Store R10, (R24) Mã tác vụ lệnh Store: 001010

d_{31}	d_{30}	d_{29}	d_{28}	d_{27}	d_{26}	d_{25}	d_{24}	d_{23}	d_{22}	d_{21}	d_{20}	d_{19}	d_{18}	d_{17}	d_{16}	d_{15}	d_{14}	d_{13}	d_{12}	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
0	0	1	0	1	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Opcode						Th/g SL					Th/g DC					Do doi 16 bit																

Add R4, R15, R11 Mã tác vụ lệnh Add: 011010

d_{31}	d_{30}	d_{29}	d_{28}	d_{27}	d_{26}	d_{25}	d_{24}	d_{23}	d_{22}	d_{21}	d_{20}	d_{19}	d_{18}	d_{17}	d_{16}	d_{15}	d_{14}	d_{13}	d_{12}	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
0	1	1	0	1	0	0	0	1	0	0	0	1	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Opcode						Th/g Dich					Th/g Nguon 1					Th/g Nguon 2					Opcode mo rong										1

Sub R3, R17, # -1500 Mã tác vụ lệnh Sub: 011011

d_{31}	d_{30}	d_{29}	d_{28}	d_{27}	d_{26}	d_{25}	d_{24}	d_{23}	d_{22}	d_{21}	d_{20}	d_{19}	d_{18}	d_{17}	d_{16}	d_{15}	d_{14}	d_{13}	d_{12}	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
0	1	1	0	1	1	0	0	0	1	1	1	0	0	0	1	1	1	1	1	1	0	1	0	0	0	1	0	0	1	0	0
Opcode						Th/g Dich					Th/g Nguon 1					So co dau 16 bit																