HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÀI GIẢNG MÔN

KIẾN TRÚC MÁY TÍNH

CHƯƠNG 3 – TẬP LỆNH MÁY TÍNH, BỘ NHỚ PHÂN CẤP, PIPELINE, STORAGES

Giảng viên:

Điện thoại/E-mail:

Bộ môn: Khoa học máy tính - Khoa CNTT1

NỘI DUNG

Chương 3: Xử lý xen kẽ dòng mã lệnh và bộ nhớ Cache

3.1 Khuôn dạng lệnh và quá trình xử lý lệnh (Kiến trúc tập lệnh)

3.2 Cơ chế xử lý xen kẽ dòng mã lệnh (Kỹ thuật đường ống-Pipeline)

- Các vấn đề pipeline: xung đột dữ liệu (data hazards), xung đột rẽ nhánh (branch hazards), và xung đột tài nguyên (resource hazards)
- *Xử lý siêu xen kẽ dòng mã lệnh (hyper-pipeline)*

3.3 Bộ nhớ Cache:

- Nguyên lý và tổ chức
- Chính sách thay thế dòng và hiệu năng bộ nhớ Cache

3.4 Công nghệ lưu trữ dữ liệu RAID

- SAN, NAS, Object Storage

NỘI DUNG KIẾN TRÚC TẬP LỆNH

1. Giới thiệu về tập lệnh máy tính
2. Dạng và các thành phần của lệnh (Khuôn dạng lệnh)
3. Địa chỉ / toán hạng của lệnh (Các loại toán hạng của lệnh)
4. Các chế độ địa chỉ
5. Các dạng lệnh thông dụng (Phân loại lệnh)
6. Câu hỏi ôn tập

3.1 Giới thiệu về tập lệnh máy tính

- Lệnh máy tính (computer instruction):
  - Là một từ nhị phân (binary word);
  - Mỗi lệnh được gán một nhiệm vụ cụ thể;
  - Lệnh được lưu trữ trong bộ nhớ
  - Lệnh được đọc (fetch) từ bộ nhớ vào CPU để giải mã và thực hiện.
- Tập lệnh gồm nhiều lệnh có thể được chia thành một số nhóm theo chức năng:
  - Chuyển giao dữ liệu (data transfer)
  - Tính toán số học và logic (computational)
  - Chuyển điều khiển (Điều kiện & rẽ nhánh) (conditonal and branching)
  - Xử lý (thao tác) bit (Set/Clear)
  - Thao tác chuỗi....

3.1 Giới thiệu về tập lệnh máy tính

- Việc thực hiện lệnh có thể được chia thành các pha (phase) hay giai đoạn (stage). Mỗi lệnh có thể được thực hiện theo 5 giai đoạn:
  - Đọc lệnh (Instruction fetch - IF): lệnh được đọc từ bộ nhớ về CPU;
  - Giải mã (Instruction decode - ID): CPU giải mã lệnh;
  - Thực hiện (Instruction execution – EX): CPU thực hiện lệnh;
  - Lấy dữ liệu trong bộ nhớ (MEM)
  - Lưu kết quả (Write back - WB): kết quả thực hiện lệnh (nếu có) được lưu vào bộ nhớ.

3.1 Giới thiệu về tập lệnh máy tính

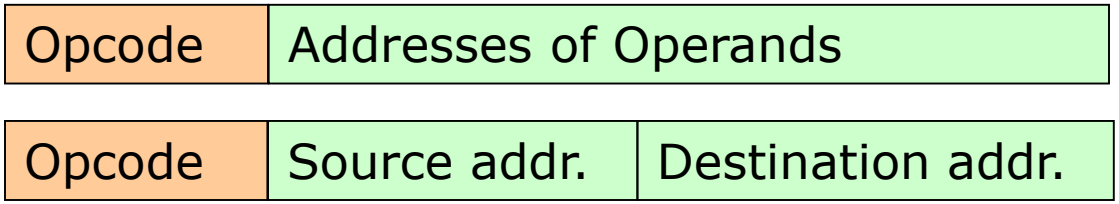
- Chu kỳ thực hiện lệnh (Instruction execution cycle): là khoảng thời gian mã CPU thực hiện xong một lệnh:
  - Một chu kỳ thực hiện lệnh có thể gồm một số giai đoạn thực hiện lệnh;
  - Một giai đoạn thực hiện lệnh có thể gồm một số chu kỳ máy;
  - Một chu kỳ máy có thể gồm một số chu kỳ đồng hồ.

3.1 Giới thiệu về tập lệnh máy tính

- Chu kỳ thực hiện lệnh có thể gồm các thành phần sau:
  - Chu kỳ đọc lệnh
  - Chu kỳ đọc bộ nhớ (dữ liệu)
  - Chu kỳ ghi bộ nhớ (dữ liệu)
  - Chu kỳ đọc thiết bị ngoại vi
  - Chu kỳ ghi thiết bị ngoại vi
  - Chu kỳ bus rỗi.

3.2 Dạng và các thành phần của lệnh

- Dạng tổng quát của lệnh gồm 2 thành phần chính:
  - Mã lệnh (Opcode - operation code): mỗi lệnh có mã lệnh riêng
  - Địa chỉ của các toán hạng (Addresses of Operands): mỗi lệnh có thể gồm một hoặc nhiều toán hạng. Có thể có các dạng địa chỉ toán hạng sau:
    - 3 địa chỉ (3 toán hạng)
    - 2 địa chỉ (2 toán hạng)
    - 1 địa chỉ (1 toán hạng)
    - 1,5 địa chỉ
    - 0 địa chỉ (không toán hnagj)



3.3 Địa chỉ / toán hạng của lệnh

- Toán hạng 3 địa chỉ (3 toán hạng):
  - Dạng:
    - opcode addr1, addr2, addr3
    - Mỗi địa chỉ addr1, addr2, addr3 tham chiếu đến một ô nhớ hoặc một thanh ghi.
  - Ví dụ:

    ADD #10, #20, R1; R1= 10 + 20

    ADD R_1, R_2, R_3; $R_1 + R_2 \rightarrow R_3$

    R_1 cộng với R_2, kết quả gán vào R_3.

    R_i là thanh ghi của CPU.

    ADD (A), (B), (C); M[A]+M[B] → M[C]

    A, B, C là các ô nhớ.

3.3 Địa chỉ / toán hạng của lệnh

- Toán hạng 2 địa chỉ:
  - Dạng:
    - opcode addr1, addr2
    - Mỗi địa chỉ addr1, addr2 tham chiếu đến một ô nhớ hoặc một thanh ghi.
  - Ví dụ:

    ADD R_1, R_2; $R_1 + R_2 \rightarrow R_2$

    R_1 cộng với R_2, kết quả gán vào R_2.

    R_i là thanh ghi của CPU.

    ADD (A), (B); M[A]+M[B] → M[B]

    A, B là các ô nhớ.

3.3 Địa chỉ / toán hạng của lệnh

- Toán hạng 1 địa chỉ:
  - Dạng:
    - opcode addr1
    - Địa chỉ addr1 tham chiếu đến một ô nhớ hoặc một thanh ghi.
    - Ở dạng 1 địa chỉ, thanh ghi R_{acc} (Accumulator) được sử dụng như địa chỉ addr2 trong dạng 2 địa chỉ.
  - Ví dụ:

    ADD R_1; R_1 + R_{acc} → R_{acc}

    R_1 cộng với R_{acc}, kết quả gán vào R_{acc}.

    R_1 là thanh ghi của CPU.

    ADD (A); M[A]+ R_{acc} → R_{acc}

    A là một ô nhớ.

3.3 Địa chỉ / toán hạng của lệnh

- Toán hạng 1,5 địa chỉ:
  - Dạng:
    - opcode addr1, addr2
    - Một địa chỉ tham chiếu đến một ô nhớ và địa chỉ còn lại tham chiếu đến một thanh ghi.
    - Dạng 1,5 địa chỉ là dạng hỗn hợp giữa ô nhớ và thanh ghi.
  - Ví dụ:

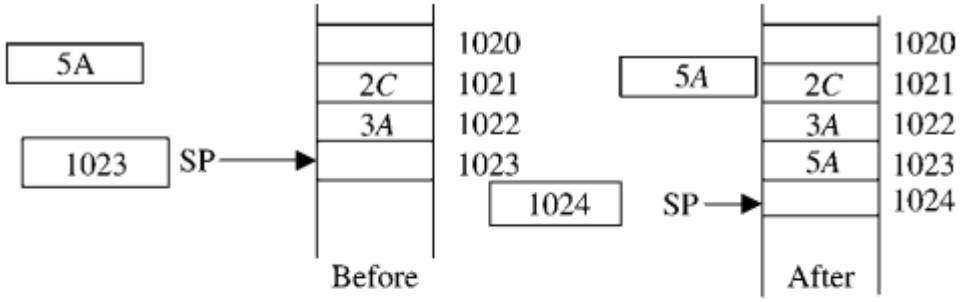
    ADD A, R_1; M[A] + R_1 → R_1

    Nội dung ô nhớ A cộng với R_1, kết quả gán vào R_1.

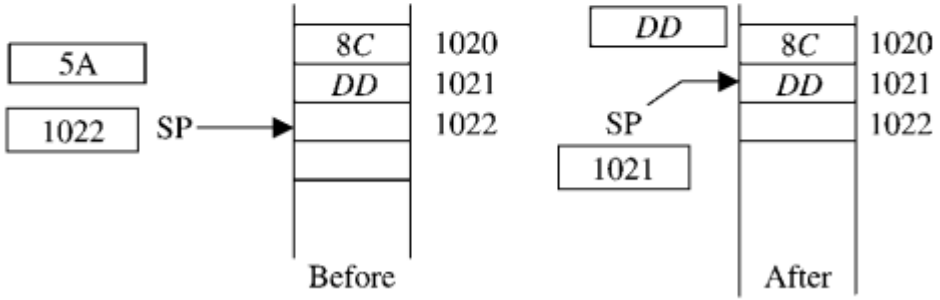
    R_1 là thanh ghi của CPU và A là một ô nhớ.

3.3 Địa chỉ / toán hạng của lệnh

- Toán hạng 0 địa chỉ: được sử dụng trong các lệnh thao tác với ngăn xếp: PUSH và POP



Stack *push* operation



Stack *pop* operation

3.4 Các chế độ địa chỉ

- Chế độ địa chỉ (Addressing modes) là phương thức CPU tổ chức và xác định vị trí dữ liệu các toán hạng của lệnh.
  - Chế độ địa chỉ cho phép CPU kiểm tra dạng và tìm các toán hạng của lệnh.
- Các chế độ địa chỉ:
  - Tức thì (Immediate)
  - Thanh ghi
  - Trực tiếp (Direct)
  - Gián tiếp qua thanh ghi (Register indirect)
  - Gián tiếp qua ô nhớ (Memory indirect)
  - Chỉ số (Indexed)
  - Cơ sở (Based)
  - Tương đối cơ sở
  - Tương đối cơ sở chỉ số

3.4.1 Các chế độ địa chỉ - Tức thì

- là chế độ định địa chỉ mà toán hạng nguồn (source operand) là một hằng số. Toán hạng đích có thể là 1 thanh ghi hoặc 1 địa chỉ ô nhớ;
- Ví dụ:

LOAD <Nguồn>, <Đích>

LOAD #1000, R_1; 1000 → R_1

Nạp giá trị 1000 vào thanh ghi R_1.

LOAD #500, (B); 500 → M[B]

Nạp giá trị 500 vào ô nhớ B.

3.4.2 Các chế độ địa chỉ - Thanh ghi

- là chế độ định địa chỉ mà Giá trị dữ liệu toán hạng nằm trong thanh ghi;
- Ví dụ:

LOAD <Nguồn>, <Đích>

LOAD #1000, R_1; 1000 → R_1

Nạp giá trị 1000 vào thanh ghi R_1.

LOAD R_1, (B); R_1 → M[B]

Nạp giá trị của R_1 vào ô nhớ B.

3.4.3 Các chế độ địa chỉ - Trực tiếp/tuyệt đối

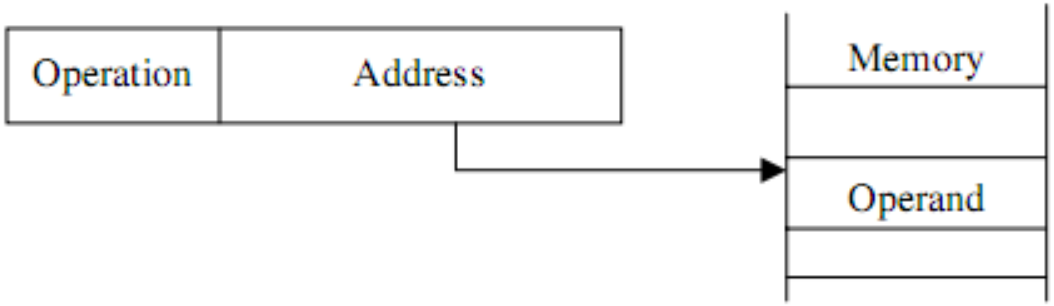
- Là chế độ định địa chỉ mà dữ liệu toán hạng nằm trong bộ nhớ, có địa chỉ là một hằng số. Toán hạng còn lại có thể là 1 thanh ghi hoặc 1 địa chỉ ô nhớ;
- Ví dụ:

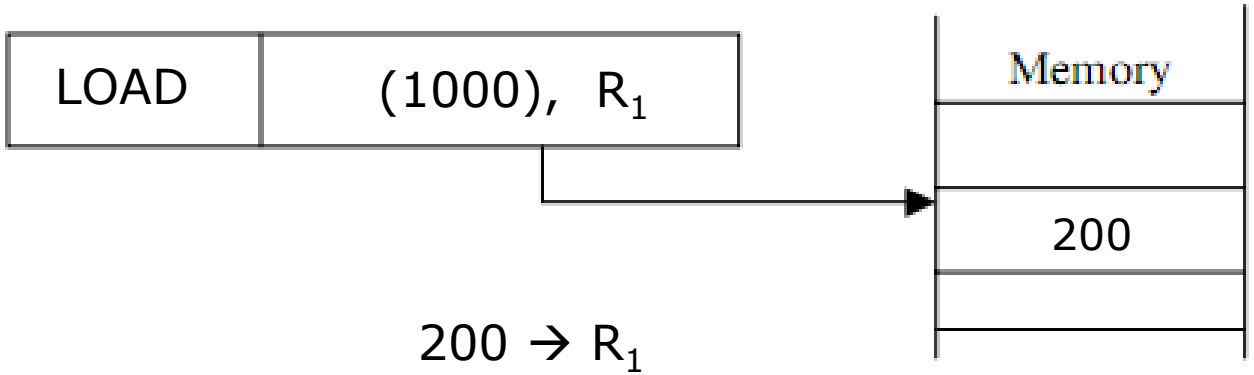
LOAD <Nguồn>, <Đích>

LOAD (1000), R_1; M[1000] → R_1

Nạp nội dung ô nhớ có địa chỉ 1000 vào thanh ghi R_1.

3.4.4 Các chế độ địa chỉ - Trực tiếp/tuyệt đối





3.4.4 Các chế độ địa chỉ - Gián tiếp thanh ghi

- Là chế độ định địa chỉ mà dữ liệu toán hạng nằm trong bộ nhớ, có địa chỉ là giá trị thanh ghi.
  - Gián tiếp qua thanh ghi:

    LOAD <Nguồn>, <Đích>

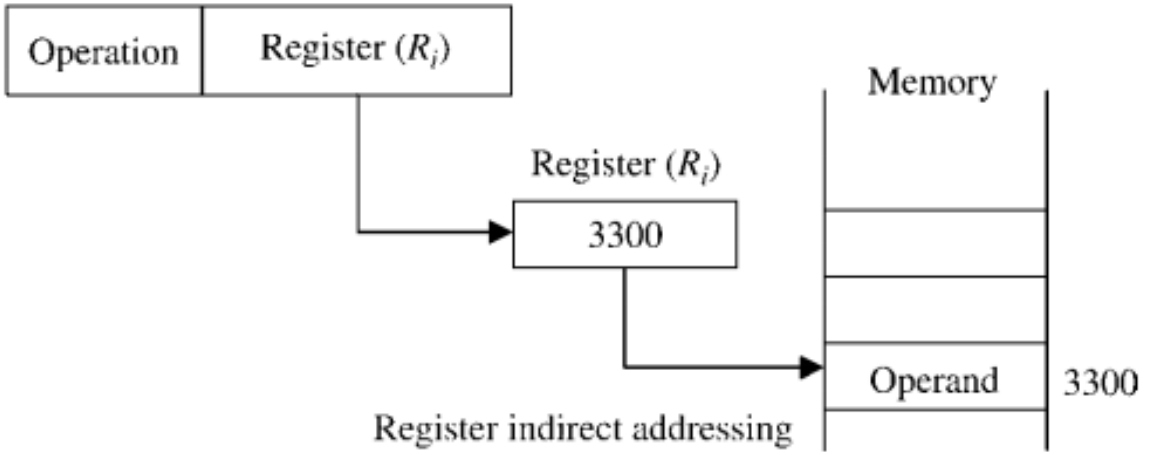
    LOAD (R_i), R_j; $M[R_i] \rightarrow R_j$

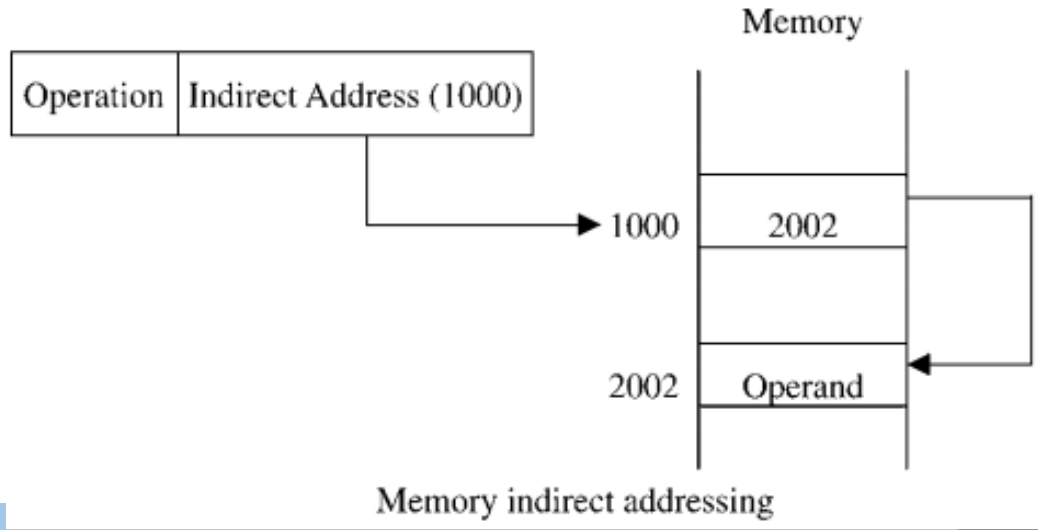
    Nạp nội dung ô nhớ có địa chỉ lưu trong thanh ghi R_i vào thanh ghi R_j.
- Gián tiếp qua ô nhớ:

  LOAD ((1000)), R_i; $M[M[1000]] \rightarrow R_i$

  Nạp nội dung ô nhớ có địa chỉ lưu trong ô nhớ 1000 vào thanh ghi R_i.

3.4.3 Các chế độ địa chỉ - Gián tiếp





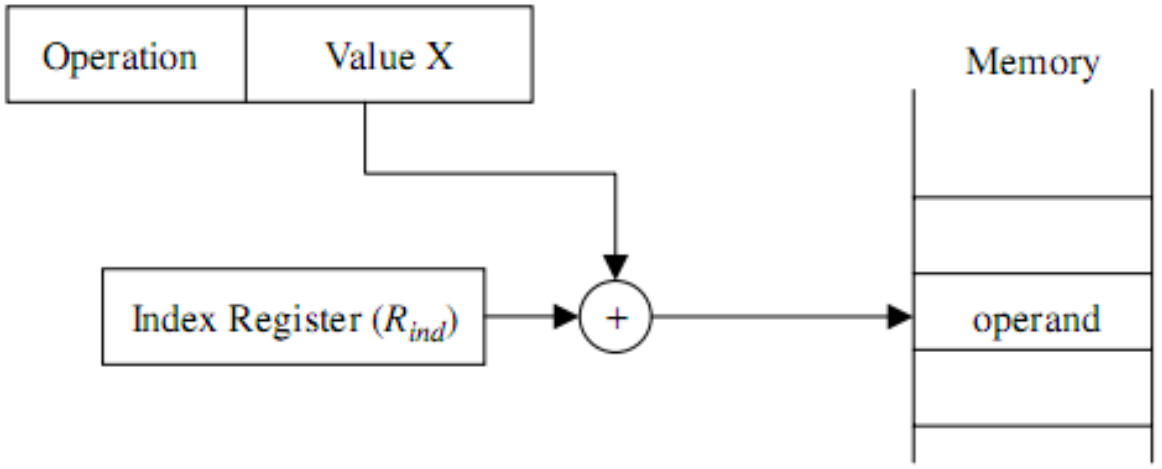
3.4.5 Các chế độ địa chỉ - Chỉ số

- Là chế độ định địa chỉ mà dữ liệu toán hạng nằm trong bộ nhớ có địa chỉ nằm trong thanh ghi chỉ số;
- Ví dụ:

  LOAD <Nguồn>, <Đích>

  LOAD (R_{ind}), R_i; $M[R_{ind}] \rightarrow R_i$

  R_{ind} là thanh ghi chỉ số.



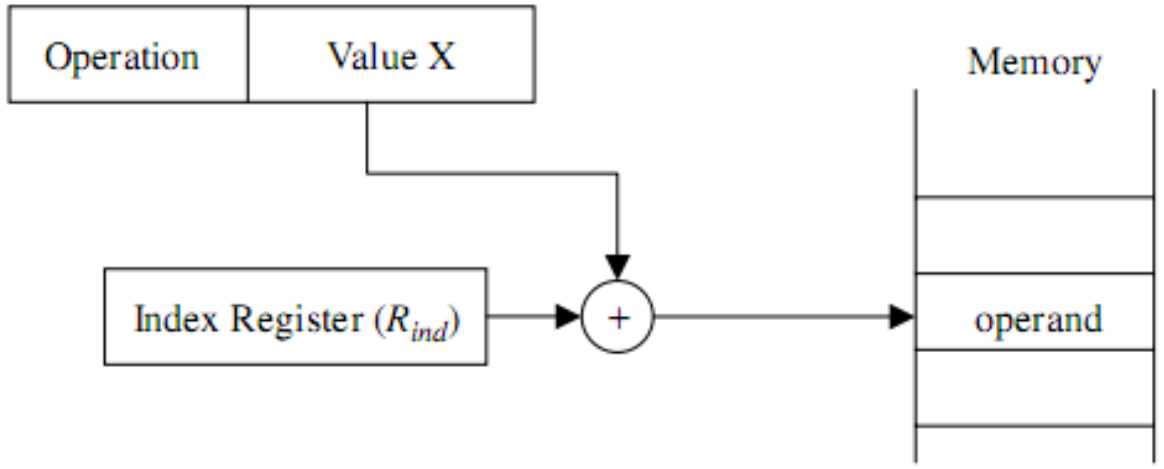
3.4.6 Các chế độ địa chỉ cơ sở

- Là chế độ định địa chỉ mà dữ liệu toán hạng nằm trong bộ nhớ có địa chỉ nằm trong thanh ghi cơ sở (Base Register);
- Ví dụ:

  LOAD <Nguồn>, <Đích>

  LOAD (R_{base}), R_i; $M[R_{base}] \rightarrow R_i$

  R_{base} là thanh ghi cơ sở

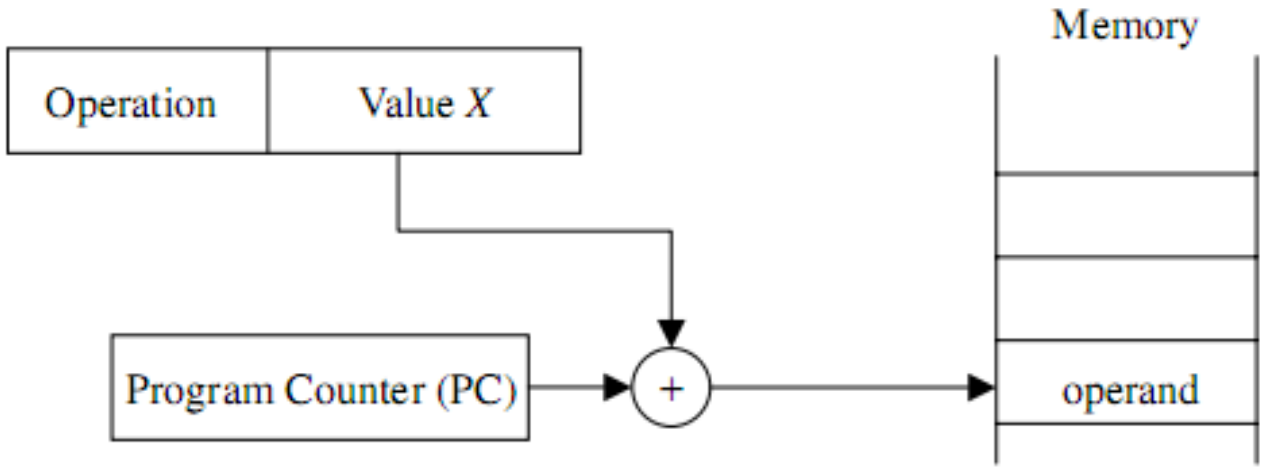


3.4.7 Các chế độ địa chỉ tương đối cơ sở

- Là chế độ định địa chỉ mà dữ liệu toán hạng nằm trong bộ nhớ, có địa chỉ là kết hợp của thanh ghi cơ sở với một hằng số;
- Ví dụ:

  LOAD <Nguồn>, <Đích>

  LOAD (R_{base} + 5), R_i; M[R_{base} + 5] → R_i

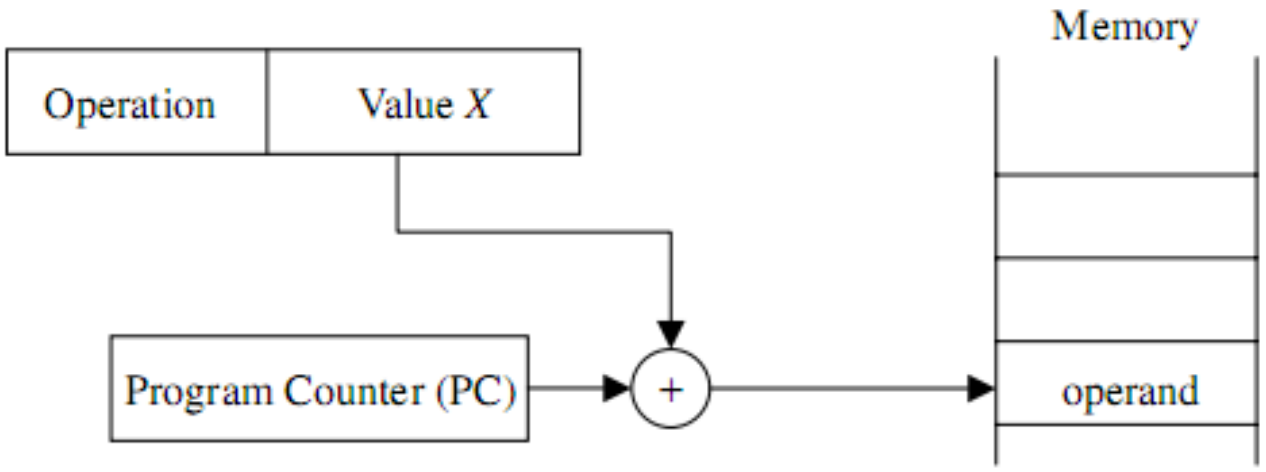


3.4.8 Các chế độ địa chỉ tương đối cơ sở chỉ số

- Là chế độ định địa chỉ mà dữ liệu toán hạng nằm trong bộ nhớ, có địa chỉ là sự kết hợp của thanh ghi cơ sở, chỉ số và một hằng số;
- Ví dụ:

  LOAD <Nguồn>, <Đích>

  LOAD (R_{index} + R_{base} + 5), R_i; M[R_{index} + R_{base} + 5] → R_i



3.5 Các dạng lệnh thông dụng (Phân loại lệnh)

- Tập lệnh máy tính có thể bao gồm một số nhóm lệnh sau:
  - Các lệnh chuyển giao dữ liệu (Data Transfer Instructions)
  - Các lệnh vào ra dữ liệu (Input/Output Instructions)
  - Các lệnh số học và logic (Arithmetic and Logical Instructions)
  - Các lệnh chuyển điều khiển chương trình (Control/Sequencing Instructions)
  - Các lệnh thao tác bít
  - Các lệnh thao tác chuỗi

3.5.1 Các lệnh vận chuyển/chuyển giao dữ liệu

❖ Lệnh vận chuyển dữ liệu giữa các bộ phận trong máy tính:

- Giữa các thanh ghi của CPU:

  MOVE R_i, R_j; R_i → R_j

- Giữa 1 thanh ghi của CPU register và một ô nhớ:

  MOVE (1000), R_j; M[1000] → R_j

- Giữa các ô nhớ:

  MOVE (1000), (R_j); M[1000] → M[R_j]

Máy tính: MOVE, PUSH, POP, EXCHAGE,LOAD, STORE

Máy tính 8086: MOV, PUSH, POP, XCHG, LOD, STO

3.5.1 Các lệnh vận chuyển dữ liệu

- Một số lệnh chuyển dữ liệu thông dụng:
  - MOVE: chuyển dữ liệu giữa thanh ghi – thanh ghi, ô nhớ - thanh ghi và ô nhớ - ô nhớ.
  - LOAD: nạp nội dung 1 ô nhớ vào 1 thanh ghi
  - STORE: lưu nội dung 1 thanh ghi ra 1 ô nhớ
  - PUSH: đẩy dữ liệu vào ngăn xếp
  - POP: lấy dữ liệu ra khỏi ngăn xếp

3.5.2 Các lệnh vào ra dữ liệu

- Các lệnh vào ra (I/O instructions) được sử dụng để vận chuyển dữ liệu giữa máy tính và các thiết bị ngoại vi;
- Các thiết bị ngoại vi giao tiếp với máy tính thông qua các cổng chuyên dụng (dedicated ports). Mỗi cổng được gán một địa chỉ;
- Hai lệnh vào ra cơ bản:
  - INPUT: sử dụng để chuyển dữ liệu từ thiết bị vào (input devices) đến CPU;
  - OUTPUT: sử dụng để chuyển dữ liệu từ CPU đến thiết bị ra (output devices).
- Máy tính 8086: In, Out

3.5.3 Các lệnh tính toán số học & logic

- Các lệnh tính toán số học & logic được sử dụng để thực hiện các thao tác tính toán trên nội dung các thanh ghi và / hoặc nội dung các ô nhớ.
- Ví dụ:

ADD R1, R2, R3; $R_1 + R_2 \rightarrow R_3$

INCREASE R1

SUBSTRACT R1, R2, R3; $R_1 - R_2 \rightarrow R_3$

DECREASE R2

ADD (A), (B), (C); M[A]+M[B] → M[C]

Máy tính 8086: Số học-ADD, SUB, MUL, DIV, INC, DEC

Logic-AND, OR, XOR, NOT, SHL, SHR, ROL, ROR

3.5.3 Các lệnh tính toán số học & logic

- Các lệnh tính toán số học thông dụng:
  - ADD: cộng 2 toán hạng
  - SUBSTRACT: trừ 2 toán hạng
  - MULTIPLY: nhân 2 toán hạng
  - DIVIDE: chia số học
  - INCREMENT: tăng 1
  - DECREMENT: giảm 1

3.5.2 Các lệnh tính toán số học & logic

- Các lệnh logic thông dụng:
  - NOT: phủ định
  - AND: và
  - OR: hoặc
  - XOR: (hoặc loại trừ) Cộng mô đun 2
  - COMPARE: so sánh
  - SHIFT LEFT/ RIGHT: dịch
  - ROTATE LEFT/ RIGHT: quay

3.5.4 Các lệnh chuyển điều khiển chương trình

- Các lệnh điều khiển chương trình được sử dụng để thay đổi trật tự thực hiện các lệnh khác trong chương trình:
  - CONDITIONAL BRANCHING (CONDITIONAL JUMP): các lệnh nhảy/ rẽ nhánh có điều kiện
    - JUMP IF <đk> Nhãn
    - JUMP IF <CF=1> Cong
  - UNCONDITIONAL BRANCHING (JUMP): các lệnh nhảy/ rẽ nhánh không điều kiện
    - JUMP Nhãn
  - CALL and RETURN: lệnh gọi thực hiện và trở về từ chương trình con.

3.5.4 Các lệnh chuyển điều khiển chương trình

- Các lệnh điều khiển chương trình được sử dụng để thay đổi trật tự thực hiện các lệnh khác trong chương trình:
  - CALL and RETURN: lệnh gọi thực hiện và trở về từ chương trình con.
- Một trong các đặc tính của các lệnh này là chúng làm thay đổi nội dung của bộ đếm chương trình PC.
- Các lệnh điều khiển chương trình sử dụng các cờ của ALU để xác định điều kiện rẽ nhánh / nhảy.

3.5.3 Các lệnh điều khiển chương trình

❖ Các lệnh điều khiển chương trình thông dụng:

- BRANCH-IF-CONDITION: chuyển đến thực hiện lệnh ở địa chỉ mới nếu điều kiến là đúng
- JUMP: chuyển đến thực hiện lệnh ở địa chỉ mới
- CALL: chuyển đến thực hiện chương trình con
- RETURN: trở về (từ chương trình con) thực hiện tiếp chương trình gọi.

VXL 8086

- Rẽ nhánh: JMP, JC
- Lặp: LOOP
- Gọi chương trình con: CALL, kết thúc chương trình con RET

3.5.3 Các lệnh điều khiển chương trình

```
        LOAD                #100, R1
Loop:   ADD                 (R2) + , R0
        DECREMENT           R1
        BRANCH-IF-GREATER-THAN  Loop
```

Lặp đến khi $R_1 = 0$

3.5.3 Các lệnh điều khiển chương trình

STT	Lệnh	Ý nghĩa lệnh	CĐĐC t/h nguồn	CĐĐC t/h đích	R1	R2
1	LOAD #100, R1	R1 =100	Tức thì	Thanh ghi	100	-
2	Loop: ADD (R2), R0	R0 = R0 +(R2)	Gián tiếp thanh ghi	Thanh ghi	100, 99, ...,1	-
3	DEC R1	R1 = R1-1	Thanh ghi	Thanh ghi	99, 98, ...,0	-
4	BRANCH IF Loop	Nếu R1>0 thì quay về Loop				-
		R0 = R0 + 100*(R2)			0	-

3.5.5 Ví dụ lập trình

	CLEAR R_0;	$R_0 \leftarrow 0$
	MOVE # 100, R_1;	$R_1 \leftarrow 100$
	CLEAR R_2;	$R_2 \leftarrow 0$
LOOP:	*ADD 1000*(R_2), R_0;	$R_0 \leftarrow R_0 + M\ (1000 + R_2)$
	INCREMENT R_2;	$R_2 \leftarrow R_2 + 1$
	DECREMENT R_1;	$R_1 \leftarrow R_1 - 1$
	BRANCH-IF > 0 *LOOP*;	*GO TO LOOP if contents of* $R_1 > 0$
	STORE R_0, *2000*;	$M(2000) \leftarrow R_0$

Đoạn chương trình cộng nội dung của 100 ô nhớ kề nhau bắt đầu từ địa chỉ 1000. Kết quả lưu vào ô nhớ có địa chỉ 2000.

3.6 Câu hỏi ôn tập

1. Khái niệm lệnh và tập lệnh? Chu kỳ lệnh và các giai đoạn thực hiện lệnh.
2. Dạng lệnh và các dạng địa chỉ toán hạng
3. Khái niệm chế độ địa chỉ và các chế độ địa chỉ
4. Một số dạng lệnh thông dụng.