**Kiến thức lý thuyết:**

* Dựa vào số lượng các toán hạng, các lệnh đưa chia thành 5 loại (chú ý rằng trong 1 bộ vi xử lý chỉ có thể xử lí 1 loại toán hạng trong toàn chương trình):
* Toán hạng 3 địa chỉ

+ Khuôn dạng: OPCODE Addr1, Addr2, Addr3

+ Mỗi địa chỉ Addr1, Addr2, Addr3 tham chiếu đến 1 ô nhớ hoặc 1 thanh ghi.

* Toán hạng 2 địa chỉ

+ Khuôn dạng: OPCODE Addr1, Addr2

+ Mỗi địa chỉ Addr1, Addr2, Addr3 tham chiếu đến 1 ô nhớ hoặc 1 thanh ghi.

* Toán hạng 1.5 địa chỉ

+ Khuôn dạng: OPCODE Addr1, Addr2

+ 1 địa chỉ tham chiếu tới 1 ô nhớ và địa chỉ còn lại tham chiếu tới 1 thanh ghi.

* Toán hạng 1 địa chỉ

+ Khuôn dạng: OPCODE Addr

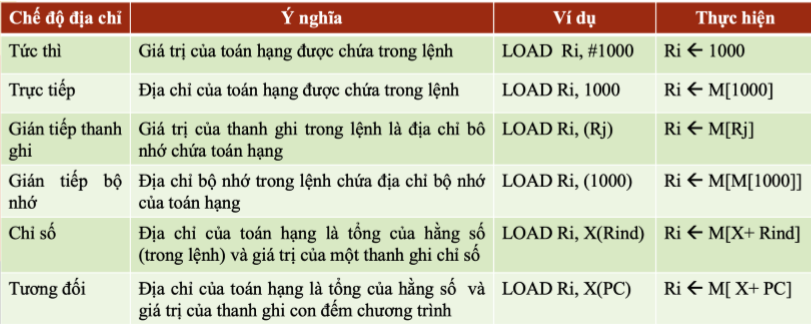
+ Địa chỉ Addr tham chiếu đến 1 ô nhớ hoặc 1 thanh ghi.

+ Khuôn dạng này sử dụng Racc mặc định cho địa chỉ thứ 2.

* Toán hạng 0 địa chỉ.

+ Khuôn dạng: OPCODE Addr

+ Được thực hiện trong các lệnh mà thực hiện các thao tác ngăn xếp (PUSH, POP)

* Một số quy ước về lệnh dạng CPU:
* Ri: là các thanh ghi của CPU
* Racc: là thanh ghi Accumulation - tích luỹ / Tổng
* (Ri): Nội dung ô nhớ có địa chỉ được lưu trong thanh ghi Ri
* (100): Nội dung ô nhớ có địa chỉ được lưu trong ô nhớ 100
* A, B, C, 1000 là địa chỉ các ô nhớ
* M[100]: tham chiếu đến nội dung ô nhớ thứ 100
* #100: Hằng số 100
* Các chế độ địa chỉ:
* Lệnh vận chuyển dữ liệu:
* MOVE: chuyển dữ liệu giữa các thanh ghi và 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/POP: đẩy/lấy dữ liệu vào/ra ngăn xếp
* Lệnh số học:
* 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
* CLEAR: đưa toán hạng về giá trị 0
* Lệnh logic:
* NOT: phủ định
* AND: và
* OR: hoặc
* XOR: hoặc loại trừ
* COMPARE: so sánh
* SHIFT: dịch
* ROTATE: quay
* Lệnh điều khiển/ tuần tự:
* JUMP: chuyển đến thực hiện lệnh ở địa chỉ mới
* BRANCH-IF-EQUAL: chuyển đến thực hiện lệnh ở địa chỉ mới nếu kết quả của phép toán = 0
* BRANCH-IF-GREATER-THAN: chuyển đến thực hiện lệnh ở địa chỉ mới nếu kết quả của phép toán > 0
* CALL: chuyển đến thực hiện chương trình con
* RETURN: trở về (từ chương trình con) thực hiện tiếp chương trình

**Bài tập:**

1. Cho đoạn lệnh sau:

MOVE R0, #100; R0 <- 100

CLEAR R1; R1 <- 0

CLEAR R2; R2 <- 0

LAP:

ADD R1, 2000(R2); R1 <- R1 + M[2000 + R2]

DECREMENT R2; R2 <- R2 – 1

DECREMENT R0; R0 <- R0 – 1

BRANCH\_IF>0 LAP; Quay lại nhãn LAP nếu R0 > 0

STORE 3000, R1; M[3000] <- R1

1. Hãy giải thích ý nghĩa của từng lệnh
2. Chỉ ra chế độ địa chỉ của từng lệnh (đối với các lệnh có 2 toán hạng)
3. Đoạn lệnh trên thực hiện công việc gì?

LG

1. Đã được comment làm luôn trên đề
2. Lệnh MOVE R0, #100: Chế độ địa chỉ tức thì

Lệnh ADD R1, 2000(R2): Chế độ địa chỉ chỉ số

Lệnh STORE 3000, R1: Chế độ địa chỉ trực tiếp

1. Đoạn lệnh trên sẽ cộng nội dung 100 ô nhớ từ ô 2000 đến ô 1901, sau đó lưu kết quả vào ô nhớ 3000.
2. Biết R0 = 1500, R1 = 4500, R2 = 1000, M[1500] = 3000, M[4500] = 500.

ADD R2, (R0);

SUBSTRACT R2, (R1);

MOVE 500(R0), R2;

LOAD R2, #5000;

STORE 100(R2), R0;

1. Chỉ rõ chế độ địa chỉ của từng lệnh.
2. Hãy chỉ ra giá trị của thanh ghi và tại vị trí trong bộ nhớ qua mỗi lệnh thực hiện.

LG

1. Lệnh ADD R2, (R0): Chế độ địa chỉ gián tiếp thanh ghi.

Lệnh SUBSTRACT R2, (R1): Chế độ địa chỉ gián tiếp thanh ghi.

Lệnh MOVE 500(R0), R2: Chế độ địa chỉ chỉ số.

Lệnh LOAD R2, #5000: Chế độ địa chỉ tức thì.

Lệnh STORE 100(R2), R0: Chế độ địa chỉ chỉ số.

1. ADD R2, (R0); R2 <- R2 + M[R0] R2 = 4000

SUBSTRACT R2, (R1); R2 <- R2 – M[R1] R2 = 4000 – 500 = 3500

MOVE 500(R0), R2; M[500 + R0] <- R2 M[2000] = 3500

LOAD R2, #5000; R2 <- 5000 R2 = 5000

STORE 100(R2), R0; M[100 + R2] <- R0 M[5100] = 1500

1. Cho 1 mảng gồm 10 số, được lưu trữ liên tiếp nhau trong bộ nhớ, bắt đầu từ vị trí ô nhớ 1000. Viết đoạn chương trình tính tổng các số **lớn hơn** **1** trong mảng đó và lưu kết quả vào ô nhớ 2000. Biết rằng mỗi ô nhớ lưu trữ 1 phần tử trong mảng.

Chương trình:

MOVE R0, #10;

CLEAR R1;

MOVE R2, #1000;

LAP:

COMPARE (R2), 1;

JUMP\_IF\_LOWER\_EQUAL THAN NEXT;

ADD R1, (R2);

NEXT:

INCREMENT R2;

DECREMENT R0;

BRANCH\_IF > 0 LAP;

STORE 2000, R1;