

THÔNG QUAN VỀ MÁY TÍNH VÀ BỘ XỬ LÝ

Các thế hệ: máy tính ra đời 1642-1940.

+ bóng đèn chân không (xem như các bit) 1946-1957, máy tính đầu tiên IBM 700 (dòng thinkpad hiện nay), hệ thống ENIAC, tốc độ 40.000 ops/s.

+ transistor 1958-1964, IBM 7094, tốc độ x5, là bước đột phá, ra đời ngôn ngữ Fortran.

+ mạch tích hợp / bán dẫn 1965-1971, IBM 360, 3000 devices/chip, tốc độ x5.

+ mạch tích hợp mật độ siêu cao LSI/ vi xử lý 1972-đến nay, hơn 100tr devices/chip, tốc độ x10.

Kiến trúc Von-Neumann: dữ liệu và chương trình chứa trong bộ nhớ ghi, bộ nhớ được định địa chỉ cho các ngăn nhớ /không phụ thuộc vào nội dung của chúng, các lệnh được thực hiện tuần tự; gồm 5 thành phần: CPU, RAM, ROM, Input, Output.

Quy luật Moore: số lượng transistor (tích hợp trong một IC/chip hoặc trên mỗi đơn vị inch²) tăng gấp đôi mỗi 18 tháng 24 tháng.

Thành phần cơ bản của máy tính (đế bàn): màn hình, bộ phận cho CPU, chân cầm ATA, RAM, khe cắm ổ cứng, nguồn điện, ổ đĩa cứng, bàn phím, chuột.

Bộ xử lý: xử lý các lệnh máy, gồm khối điều khiển (điều khiển xử lý ALU và data trên register), ALU, register, (internal bus - kết nối CU, ALU và register).

Các trúc chính (5): bộ xử lý (CPU), hệ thống nhớ (main memory/ trong - ngoài), hệ thống kết nối (bus), I/O.

Wafer (đế chip): tấm silicon để cấy các vật liệu -> tạo vi mạch, kích thước 1inch(25.4mm) ~ 7.9inch (200mm)

Chip: là mạch tích hợp trên wafer được để xử lý các công việc, có thể chứa hàng chục triệu transistor, gồm 4/8/16/32/64 bit.

Chipset là tập hợp nhiều chip trên wafer, thông dụng là **CPU, GPU** (đơn vị xử lý đồ họa), **RAM** (bộ nhớ truy cập tức thời phục vụ CPU), **bàn cầu bắc** (tích hợp trên mainboard hỗ trợ truyền thông tin cho CPU/RAM, nằm kế CPU), **bàn cầu nam** (quản lý thiết bị ngoại vi như HDD, mouse, keyboard, nằm cuối mainboard).

Chức năng cơ bản của máy tính: lưu trữ + xử lý + trao đổi dữ liệu + điều khiển. Thực hiện theo trình tự: nhận -> xử lý -> xuất thông tin.

Các hoạt động máy tính gồm: thực hiện chương trình, nhận, vào/ra.

Ngắt: khả năng tạm ngừng chương trình để thực thi chương trình khác, xảy ra khi thiết bị phần cứng hay chương trình cần sự giúp đỡ của CPU nó sẽ gửi đi tín hiệu hoặc lệnh đến bộ vi xử lý.

Các loại ngắt trong máy tính: ngắt cứng do các tín hiệu INTR là ngắt chân được hoặc ngắt không chắn được NMI đòi hỏi CPU thực hiện ngay khi có yêu cầu (sự cố điện/ lỗi bộ nhớ), mức độ ưu tiên cao nhất; **ngắt mềm** do lệnh INT (ROM-BIOS); **ngắt ngoại lệ** do lệnh của CPU như chia 0, flag.

BIỂU DIỄN SỐ NGUYÊN

Số lượng dấu: n bit biểu diễn $2^{n-1}+1$ đến $2^{n-1}-1$, phức tạp cho máy tính phân biệt số 0.

Số bù 1: giới hạn, hạn chế giống lượng dấu.

Số bù 2: giới hạn $2^{n-1} > 2^{n-1}-1$, khắc phục vấn đề có 2 số 0 của lượng dấu và bù 1.

Số bias: n bit thì k $= 2^{n-1}-1$, N>k thì +, < thì -

AND: với 0 ra 0, với 1 ra chính nó, bit nào cần giữ lại giá trị thì AND 1, còn lại AND 0, dùng làm mask, chuyển ký tự thường thành hoa, chuyển đại bit thành đại nhỏ hơn.

OR: với 0 bằng chính nó, với 1 ra 1, bit cần bật lên thì OR 1, còn lại OR 0, chuyển từ số sang ký tự số, ký tự hoa sang thường, chuyển đại bit thành đại bit lớn hơn.

XOR: x XOR x = 0, dùng để đảo bit, XOR 0 bằng chính nó, x XOR 1 = not(x); nếu số 0 là số lẻ, -> nhận biết XOR. **SHL/SHR:** dịch logic, dịch hướng nào thì bit ngoài cùng hướng ngược lại bằng 0. Dịch trái thì kết quả nhân 2^k , dịch phải thì chia 2^k .

SAR: dịch phải số học, bit phải nhất bằng bit dấu bán dư, kết quả chia 2^k .

Một số khái niệm đặc biệt:

(*x SHR i*) là: lấy giá trị bit thứ i.

(*1 SHL i*) OR x = (*1 SHR (N-1-i)*) OR x

= (*x SHL i*) AND 1: bật bit I bằng 1.

NOT (*1 SHL i*) AND x: tắt bit thứ i, i = 0.

(*1 SHL i*) XOR x: đảo bit thứ i.

Phương toán cộng/trừ: bù 1 nếu dư bit MSB thì cộng tiếp, bù 2 dư thì bỏ.

Nhận biết tràn số: với số không dấu thì nhờ vào nhưng không nhớ ra, ngược lại hoặc có nhớ ra khỏi bit cao nhất; với số có dấu thì kiểm tra dấu của kết quả, sai khi cộng hai số cùng dấu cho ra số khác dấu,

Thuật toán nhân không dấu: C (1bit) = 0 và A = 0. Khởi tạo dãy CAQ. Với Q là số nhân, M là số bị nhân. Lập: xét bit cuối của Q nếu = 1 thì CA=A+M, sau đó dịch phải dãy CAQ. Res = AQ.

Thuật toán Booth - nhân số bù 2: A = 0, Q₁ (1 bit)=0. Khởi tạo dãy AQQ,M. Lập: xét Q₀Q₁ nếu = 01 thì

A=A-M, nếu = 10 thì A=A-M, sau đó dịch phải dãy AQQ. Kết quả = AQ.

Thuật toán chia số ko dấu: A = 0, Q là số bị chia, M là số chia. Khởi tạo dãy AQ. Lập: dịch trái AQ, A = A-M nếu A >=0 thì Q_i=1, ngược lại Q_i=0 và A = A+M. Res = Q, dư=A.

Thuật toán chia số bù 2: thực hiện như phép chia không dấu, nếu số bị chia và số chia khác dấu thì đổi dấu kết quả.

BIỂU DIỄN SỐ THỰC

Giới hạn số chấm tính: n bit thì phần thập phân nhỏ nhất có thể biểu diễn là 2^{-n} .

Theo chuẩn IEEE 754/85: có 3 dạng biểu diễn số thực + Single: độ dài 32bit với các trường SEM là 1+8+23.

X = (-1)^S * 1.M. R^A (E-127)

+ Double: độ dài 64bit với các trường 1+11+52.

X = (-1)^S * 1.M. R^A (E-1023)

+ Double extended: 80bit với các trường 1+15+64.

X = (-1)^S * 1.M. R^A (E-16383)

Số chấm động: 1 bit dấu (sign), 8 bit mũ (E) và 23 bit định trị (S) -> giá trị = $S * 2^E$.

Các số đặc biệt: số 0 (mũ = 0, trị = 0), số vô cùng (mũ = 0, trị #0), số vô cùng (mũ = 1, trị = 0), số bảo lưu (mũ = 1, trị #0).

Phạm vi biểu diễn:

+ Dạng chuẩn lớn nhất: 1.[23 số 1]* 2^{+127} .

x 1111 1110 1111 1111 = (2- 2^{-23})* 2^{+127} .

(tương tự cho số âm nhỏ nhất -(2- 2^{-23})* 2^{+127})

+ Dạng chuẩn nhỏ nhất: 1.[23 số 0]* 2^{-126} .

x 0000 0001 0000 0000 = 2^{-126}

+ Số dạng không chuẩn lớn nhất: 0.[23 số 1]* 2^{+127} . Tuy nhiên IEEE754 quy định là 0.[23 số 1]* 2^{+126} vì muốn tiện hơn với số dương dạng chuẩn nhỏ nhất.

x 0000 0000 1111 1111

+ Số dạng không chuẩn nhỏ nhất: 1.[22 số 0]* 2^{+127} . Tuy nhiên IEEE754 quy định là 0.[22 số 0]* 2^{-126} .

x 0000 0000 0000001 = 2^{-149} .

Trong C: phạm vi kiểu float từ 10^{-38} đến 10^{38} , kiểu double từ 10^{-308} đến 10^{308} .

Precision: số bit được sử dụng để biểu diễn 1 giá trị. **Accuracy:** độ chính xác mà một kiểu biểu diễn có thể biểu diễn 1 giá trị.

Rounding: phần cứng hỗ trợ 2 bit nhỏ hơn phần định trị giúp làm tròn kết quả.

Chuẩn IEEE làm tròn số chấm động:

+ tròn lên: 1.01 10 -> 1.10, -1.01 10 -> -1.01

+ tròn xuống: ... -> 1.01, ... -> 1.10

+ tròn về 0: bỏ giá trị 2 bit nhỏ.

+ tròn về giá trị gần nhất.

Mã Unicode: bộ mã 2byte, đa ngôn ngữ có hỗ trợ TV. **ASCII:** bộ mã 1byte do ANSI thiết kế có chữ các ký tự điều khiển truyền tin (máy in/màn hình), có chứa các ký tự ký không

MẠCH SỐ

Transistor: linh kiện điện tử làm từ chất bán dẫn dùng để khuếch đại và chuyển tín hiệu đến.

Mạch số: thiết bị điện tử kết nối các linh kiện điện tử (transistor) hoạt động ở 2 mức điện áp (cao và thấp).

Cổng logic: các linh kiện điện tử thường kết nối với nhau thành các khối cơ bản là cổng logic với giá trị luận lý 1 và 0 tương ứng với 2 mức điện áp là thể cao/thấp.

AND	Hình vẽ	Ký hiệu
OR		x+y
XOR		x⊕y
NOT		¬x hoặc x'
NAND		¬(x.y)
NOR		¬(x+y)
NXOR		¬(x⊕y)

Thiết kế logic: các cổng logic được kết nối với nhau thành các khối cơ bản, gồm:

Mạch tổ hợp: kết quả của mạch chỉ phụ thuộc vào giá trị đầu vào tại thời điểm đang xét: adder, decoder, multiplexer, ALU,...Gồm n ngõ vào, m ngõ ra (hàm luận lý của các ngõ vào), propagation delay là độ trễ giữa thời điểm tín hiệu vào ổn định và ra ổn định.

Mạch tuần tự: kết quả của mạch không chỉ phụ thuộc vào giá trị đầu vào mà còn phụ thuộc vào trạng thái tại thời điểm trước đó của mạch: mạch latch, JK, T, D,... có khả năng ghi nhớ các trạng thái trong quá khứ.

Các mạch số (mạch xử lý) được thiết kế ở mức logic sau đó dùng các kỹ thuật khác nhau để chuyển thành mạch số ở mức các linh kiện điện tử. Dùng để thiết kế counter, register, cache memory.

Thiết kế mạch tổ hợp: lập bảng chân trị, xây dựng hàm luận lý, vẽ sơ đồ, truth nghiệm.

SOP: (sum of products): đầu ra 0 nhiều hơn 1.

POS: (products of sum): đầu ra 1 nhiều hơn 0.

Luật cơ bản: not(A+B) = not(A).not(B)

A.not(A) = 0 A + not(A) = 1

1 + A = 1 A.A = A not(A.B) = not(A) + not(B)

MUX - multiplexer: (mạch dồn) chọn n ngõ trong 2^n ngõ vào để quyết định giá trị ngõ ra duy nhất. **DEMUX:** (mạch tách) chọn n để lựa chọn ngõ vào in sẽ được chuyển đến ngõ ra nào trong 2^n ngõ ra. **ALU:** được thiết kế đơn giản, cho phép lựa chọn và thực hiện các phép toán ADD, SUB, AND OR.

Mạch lật flip-flop là mạch tuần tự cơ bản nhất, có chức năng lưu trữ 1 bit nhớ.

Thanh ghi dịch: (shift register) 4 bit tạo từ 4 mạch lật R với dây bit nạp 10010000.

Kiến trúc MIPS

Lệnh máy: dãy bit mà bộ xử lý hiểu để thực thi công việc nào đó.

Bộ lệnh: (instruction set) tập hợp các lệnh mà một bộ xử lý nào đó cài đặt. Các bộ xử lý khác nhau cài đặt các bộ lệnh khác nhau. Máy tính chỉ hiểu được ngôn ngữ máy, để gọi nhớ ngôn ngữ máy -> hợp ngữ

Biểu diễn: ngôn ngữ cấp thấp, cung cấp cách thể hiện gọi nhớ cho các lệnh máy. Dùng mã địa chỉ, tên gọi (label, tên biến,...) để ghi nhớ các mã lệnh, địa chỉ, tên gọi dữ liệu hoặc lưu lệnh. Hợp ngữ là cho một bộ xử lý hoặc một dòng bộ xử lý (cùng kiến trúc).

Trình biên dịch hợp ngữ: có thể đưa vào các mở rộng của riêng mình nên chương trình viết bằng hợp ngữ sẽ mang đặc trưng riêng phụ thuộc vào assembler mà tác giả sử dụng. Có thể tồn tại nhiều assembler cho cùng 1 assembly language.

ISA: (instructor set architecture) là tập lệnh dành cho những bộ vi xử lý khác nhau có kiến trúc tương tự nhau (có thể thực thi cùng 1 chương trình). Thông dụng là 80x86 (x86) của Intel, IA-16, IA-32, IA-64, MIPS (dùng nhiều để hệ thống nhúng), PowerPC của IBM.

RISC & CISC: complex là bộ lệnh gồm nhiều lệnh từ đơn giản đến phức tạp và reduced là bộ lệnh chỉ gồm các lệnh đơn giản.

Nguyên tắc MIPS: đơn giản có quy tắc, nhỏ gọn -> xử lý nhanh, tăng tốc độ xử lý cho những trường hợp thường xuyên xảy ra, thiết kế tốt đối với sự thỏa hiệp tốt.

Cấu trúc: 32 thanh ghi (\$0-\$31), mỗi thanh ghi 4byte (1 word), 2 toán hạng nguồn, 1 toán hạng đích. Các toán hạng là địa chỉ thanh ghi để đơn giản và thao tác nhanh.

Các trúc R-format: opcode = 0

opcode	rs	rt	rd	shamt	func
6	5	5	5	5	6

opcode: mã thao tác, cho biết loại lệnh gì.

func: kết hợp với opcode để xác định lệnh làm gì, trường này không nằm sát opcode vì nếu bộ trường func có thể tạo cấu trúc mới.

shamt: chứa số bit cần dịch trong lệnh dịch, có kích thước 5 bit, biểu diễn đủ 32 thanh ghi, nếu không phải lệnh dịch thì = 0.

rs: (source) chứa địa chỉ thanh ghi nguồn 1.

rt: (target) chứa địa chỉ thanh ghi nguồn 2.

rd: (destination) chứa địa chỉ thanh ghi đích, mỗi trường chứa đủ để biểu diễn 32 reg

Lệnh hợp ngữ sơ học và luận lý:

opt opr, opr1, opr2

opt: tên thao tác (operator), opr: thanh ghi đích, opr1: reg1, opr2: reg2 hoặc hằng số.

Toán hạng thanh ghi: không có kiểu, thao tác trên thanh ghi sẽ xác định dữ liệu trong đó được để xử lý như thế nào. Ưu điểm bộ xử lý truy xuất thanh ghi nhanh nhất (1 tỉ lần trong 1 giây) vì thanh ghi nằm chung mạch với bộ xử lý. Khuyết điểm là số lượng cố định và hạn chế.

Lưu biến: 8 thanh ghi thường sử dụng là \$16-\$23, đặt tên gọi như là \$0 - \$7 (saved)

Cộng có dấu: add \$0, \$1, \$2 (bù 2)

Cộng không dấu: addu \$0, \$1, \$2

Trừ có dấu: sub \$0, \$1, \$2 (bù 2)

Trừ không dấu: subu \$0, \$1, \$2

MIPS có định 32 bit và các lệnh đơn giản để tăng tốc độ xử lý nên xây dựng lệnh có nhiều toán hạng nguồn sẽ tăng độ phức tạp cấu lệnh -> 2 thanh ghi nguồn là đủ.

Lưu tạm kết quả trung gian: 8 thanh ghi thường dùng là \$8 - \$15, dành số \$0 - \$7.

Thanh ghi zero: \$0 hay \$zero có hợp trợ phép gán và thao tác với 0. Vì sự đơn giản nên không cần thêm lệnh gán

lui \$at, \$0xABC

ori \$at, \$at, 0xCDCD

add \$0, \$0, \$at

Trần số: add/addi/subu phát hiện tràn số nhưng addu/addiu/subu thì không ->trình biên dịch sẽ chọn các lệnh số học tương ứng

(C trên MIPS dùng addu/addiu/subu).

địa chỉ 1 thanh ghi vào 1 thanh ghi, đồng thời lệnh gán sẽ chỉ dùng 2 thanh ghi -> vi phạm nguyên tắc.

Tính toán luận lý: hỗ trợ lệnh and, or, nor với toán hạng nguồn thứ 2 phải là thanh ghi. Không có lệnh not A nor 0 = not(A), các lệnh xor, nand có thể phân rã thành các lệnh nhỏ hơn nên không cần thiết.

Phương pháp: toán hạng thứ 2 phải là hằng số. **Sll \$s1, \$s2, 2** -> dịch trái luận lý \$s2 bit lưu vào \$s1, tương tự với srl, sra.

Truy xuất bộ nhớ: bộ nhớ là mảng 1 chiều các ô nhớ có địa chỉ. Để truy xuất dữ liệu trong bộ nhớ thì cần phải giữ địa chỉ ô nhớ. Thao tác với bộ nhớ cần ít nhất 1 toán hạng nguồn là 1 toán hạng đích.

Cấu trúc I-format: cấu trúc lệnh để thao tác với bộ nhớ, giảm thiểu thay đổi so với R-format. Địa chỉ dùng để truy xuất xác định bằng tổng của một thanh ghi chứa địa chỉ vùng nhớ (xem như con trỏ) và 1 số nguyên (xem như độ dời -byte).

opcode rs rt immediate

6 5 5 16

opcode: tương tự R-format, không có func.

rs: chứa địa chỉ thanh ghi nguồn 1.

rt: (target) chứa địa chỉ thanh ghi đích.

immediate: 16 bit có thể biểu diễn số nguyên từ 2^{+15} đến $2^{+15}-1$, đủ lớn để chứa giá trị độ dời (offset) từ địa chỉ trong thanh ghi cơ sở rs nhằm phụ vụ việc truy xuất bộ nhớ trong lw và sw.

Data transfer instructions: lệnh lưu trữ dữ liệu được MIPS hỗ trợ để di chuyển dữ liệu giữa thanh ghi và vùng nhớ (load-store)

Bộ xử lý nạp các dữ liệu (lệnh) vào các thanh ghi để xử lý rồi lưu trữ lại bộ nhớ.

Lệnh di chuyển dữ liệu:

opt opr, opr1 (opr2)

opt: tên thao tác, opr: thanh ghi lưu trữ nhớ, opr1: hằng số nguyên, opr2: thanh ghi chứa địa chỉ vùng nhớ

Nap: 1 từ dữ liệu bộ nhớ vào thanh ghi

lw \$t0, 12(\$0) -> nạp từ nhớ có địa chỉ (\$0+12) vào thanh ghi \$t0.

Sw: 1 từ dữ liệu thanh ghi vào bộ nhớ

sw \$t0, 12(\$0) -> lưu giá trị trong thanh ghi \$t0 vào vùng nhớ có địa chỉ (\$0+12).

Nguyên tắc Alignment Restriction: các đối tượng lưu trong bộ nhớ phải bắt đầu địa chỉ là bội số của kích thước đối tượng. Với MIPS, bội số là 4.

Nguyên tắc Bid Endian: đối với giá trị có kích thước > 1 byte thì byte cao sẽ lưu tại địa chỉ thấp (trái với Little Endian trong x86 - intel hiện nay).

Vd: 12345678

Địa chỉ: 0 1 2 3 (BIG)

3 2 1 0 (LITTLE)

Con trỏ: địa chỉ giữ số add \$t1, \$t0 thì z = x+y nhưng lw \$t2,0(\$t0) thì \$t0 chứa địa chỉ tương đương con trỏ z=x.

Nap, lưu 1 byte: loadbyte lb, storebyte sb.

Cú pháp tương tự với lw, sw. Hỗ trợ các thao tác với ký tự 1 byte (ASCII)

Vd: lw \$0, 3(\$s1) -> nạp giá trị lưu nhớ địa chỉ (\$s1+3) vào \$0, \$s1

Nap, lưu 2 từ: lưu vào byte thấp của thanh ghi, hỗ trợ các thao tác với ký tự 2 byte (Unicode); loadhalf lh, store half sh.

Các lệnh nạp, lưu sử dụng vùng nhớ nhưng các lệnh khác không dùng vì để mạch xử lý đơn giản hơn, nâng cao tốc độ bằng cách sử dụng các kỹ thuật song song.

Trình biên dịch: ảnh xạ các biến được sử dụng trong chương trình thành các thanh ghi (spilling).

Thao tác với hằng số: (I-format) được hỗ trợ do các thao tác với toán hạng hằng số (lệnh dịch, di chuyển) xuất hiện thường xuyên, giúp chương trình chạy nhanh hơn vì hằng số không cần lưu vào thanh ghi khác.

Cộng hằng số có dấu: add \$0, \$s1, -10

Cộng hằng số ko dấu: addiu \$0, \$s1, 10

Tính toán luận lý: andi, ori.

Vấn đề: giới hạn trường hằng số chỉ có 16 bit -> tăng kích thước lệnh thao tác với hằng số có cấu trúc I-format

Load Upper Immediate: lui reg, imm, đưa hằng số 16 bit vào 2 byte cao của một thanh ghi, giá trị các bit 2 byte thấp gán bằng 0.

Vd: muốn cộng 32bit 0xABABCDCD với \$t0.

lui \$at, 0xABAB

ori \$at, \$at, 0xCDCD

add \$t0, \$0, \$at

Trần số: add/addi/subu phát hiện tràn số nhưng addu/addiu/subu thì không ->trình biên dịch sẽ chọn các lệnh số học tương ứng

(C trên MIPS dùng addu/addiu/subu).

Rẻ nhánh có điều kiện: (I-format) cần 2 toán hạng nguồn để so sánh và 1 toán hạng cho biết địa chỉ cần nhảy tới.

bneq register1, register2, L1 -> reg1=reg2

bneq register1, register2, L1 -> reg1=reg2

PC-Relative Addressing: immediate là số lệnh cần nhảy hoặc để tới được nhận, xem như số có dấu cộng với địa chỉ trong PC reg, nghĩa là chứa khoảng cách so với địa chỉ nằm trong PC reg (địa chỉ đang thực hiện).

Khi đó có thể nhảy tới, lui khoảng 2^{+15} từ nhớ PC (2^{+17} bytes), đáp ứng hầu hết các yêu cầu nhảy lập (tối đa 50 hoặc ít (khoảng 3200).

Địa chỉ rẽ nhánh: nếu không thực hiện rẽ nhánh thì PC = PC+4 (địa chỉ lệnh kế tiếp) nhưng nếu có thì PC = (PC+4) + imm*4. -> lệnh rẽ nhánh là đặc trưng của MIPS, chỉ các cấu trúc thuộc RISC đơn giản và có quy tắc mới xây dựng được vì x86 các lệnh có kích thước không bằng nhau nên không thể.

Cấu trúc J-format: lệnh rẽ nhánh không điều kiện, cần chỉ 1 toán hạng cho biết địa chỉ cần nhảy tới.

opcode target address

6 26

Cấu trúc này có thể nhảy trong khoảng 2^{+26} từ nhớ (2^{+28} byte). Thông thường không cần thiết nhảy tới các từ nhớ có địa chỉ từ $2^{+27} - 2^{+28}$ nhưng nếu cần thiết thì có lệnh jr.

jalabel -> beq \$0, \$0, label

So sánh không bằng: slt reg1, reg2, reg3

Nghĩa là if (reg2 < reg3) reg1 = 1 else reg1=0

Cập slt->bne tương đương if (x<y) goto z.

Hàng số so sánh không bằng: (I-format) hữu ích với vòng lặp for: slti \$0, \$0, 1.

Lời gọi thủ tục: khi gọi thủ tục, lệnh tiếp theo được thực hiện là lệnh đầu tiên của thủ tục -> xem tên thủ tục là một nhân và lời gọi thủ tục là lệnh nhảy tới nhân này.

jal label (jump and link, J-format) lưu địa chỉ của lệnh kế tiếp vào thanh ghi

