## TỔNG QUAN VỀ MÁY TÍNH & BÔ XỬ LÝ

Các thế hệ: máy tính cơ học 1642-1940.

- + bóng đèn chân không (xem như các bit) 1946-1957, máy tính đầu tiên IBM 700 (dòng thinkpad hiện nay), hê thống ENIAC, tốc đô 40,000 ops/s.
- + transistor 1958-1964, IBM 7094, tốc đô x5, là bước đột phá, ra đời ngôn ngữ Fortran.
- + mạch tích hợp/ bán dẫn 1965-1971, IBM 360, 3000 devices/chip, tốc đô x5.

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

Kiến trúc Von-Neumann: dữ liệu và chương trình chứa trong bô đọc ghi, bô nhớ được đánh 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.

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

Thành phần cơ bản của máy tính (để bàn): màn hình, bo mạch chủ, CPU, chân cắm ATA, RAM, khe cắm mở rông, 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 CII ALII và register)

Cấu trúc chính (5): bô xử lý (CPU), hê thống nhớ (main memory/ trong - ngoài), hệ thống kết nối (bus), I, O. Wafer (để chip): tấm silicon để cấy các vật liệu -> tạo

vi mach, kích thước 1inch(25.4mm) - 7.9inch (200mm) Chip: là mạch tích hợp gắn trên wafer dùng để xử lí 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 vi xử lý đồ hoa), RAM (bô nhớ truy cập tức thời phục vụ CPU), bán cầu bắc (tích hợp trên mainboard hỗ trơ truyền thông tin cho CPU/RAM, nằm kế CPU), bán cầu nam (quản lý thiết bị ngoại vị như HDD. mousse, keyboard, nam 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, ngắt, vào/ra.

Ngắt: khẳ năng tạm ngừng chương trình để thực thi chương trình khác, xảy ra khi thiết bị phần cứng hay chương trình cần sư giúp đỡ của CPU nó sẽ gửi đi tín hiệu hoặc lệnh đến bộ vị xử lý. Các loại ngắt trong máy tính: ngắt cứng do các tín

hiệu INTR là ngắt chắn được hoặc ngắt không chắn được NMI đòi hỏi CPU thực hiện ngay khi có yêu cầu (sư cố điên/ lỗi bộ nhớ), mức độ ưu tiên cao nhất; ngắt mêm do lênh INT (ROM-BIOS); ngắt ngoại lê do lênh của CPU như chia 0, flag.

# BIỂU DIỄN SỐ NGUYÊN

Số lượng dấu: n bit biểu diễn -2<sup>n-1</sup>+1 đến 2<sup>n-1</sup>-1, phức tạp cho máy tính phân biệt số 0.

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

Số bù 2: giới han  $-2^{n-1} -> 2^{n-1} -1$ , 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 lai giá tri thì AND 1, còn lại AND 0, dùng làm mask, chuyển ký tư thường thành hoa, chuyển dãy bit thành dãy 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 dãy bit thành dãy bit lớn hơn.

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

Môt số kết quả đặc biệt: (x SHR i) AND 1: lấu giá tri bit thứ i.

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

= (x SHL i) AND 1: bât bit I bằng 1.

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

(1 SHL i) XOR x: đảo bịt thứ i. Phép toán cộng/trừ: bù 1 nếu dư bit MSB thì cộng tiếp, bù 2 dự thì bỏ.

Nhân biết tràn số: với số không dấu thì nhớ vào nhưng không nhớ ra, ngược lai 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 đấu:  $C_1(1hit) = 0$  và A = 0Khởi tạo dãy CAQ. Với Q là số nhân, M là số bị nhân. Lặp: xét bit cuối của Q nếu = 1 thì CA=A+M, sau đó dịch phải

Thuật toán Booth - nhân số bù 2: A = 0.0.1 (1 bit) = 0.Khởi tạo dãy AQQ.1M. Lặp: xét Q0Q.1 nếu = 01 thì A=A+M, nếu = 10 thì A=A-M, sau đó dịch phải dãy AOO.1. Kết quả = AO.

Thuật toán chia số ko dấu: A = 0, Q là số bị chia, M là số chia. Khởi tao dãy AQ. Lặp: dịch trái AQ, A = A-M nếu A > 0 thì  $O_0 = 1$  ngược lại  $O_0 = 0$  và A = A + M Res = 0 dự= AThuật toán chia số bù 2: thực hiện như phép chia không dấu, nếu số bị chia và số chia khác dấu thì đổi dấu

#### BIỂU DIỄN SỐ THỰC

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

Theo chuẩn IEEE 754/85: có 3 dang biểu diễn số thực + Single: độ dài 32bit với các trường SEM là 1+8+23.  $X = (-1)^S \cdot 1_i M \cdot R^{(E-127)}$ 

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

 $X = (-1)^S 1 M R^{(E-1023)}$ + Double extended: 80bit với các trường 1+15+64

 $X = (-1)^S \cdot 1, M \cdot R^(E-16383)$ Số chấm động: 1 bit dấu (sign), 8 bit mũ (E) và 23 bit  $dinh tri (S) \rightarrow giá tri = S \times 2^{E}$ .

Các số đặc biệt: số 0 (mũ = 0, trị = 0), số dạng ko chuẩn (mũ = 0, trị #0), số vô cùng (mũ = 1, trị = 0), số báo lỗi  $(m\tilde{u} = 1, tri #0).$ 

#### Pham vi biểu diễn:

+ Dang chuẩn lớn nhất: 1,[23 số 1]\*2127.

x 1111 1110 1111 1111 .... = (2-2-23)\*2127. (tương tư cho số âm nhỏ nhất -(2-2-23)\*2127)

+ Dang chuẩn nhỏ nhất: 1.[23 số 0]\*2-126. x 0000 0001 0000 0000 ..... = 2-126

+ Số dang không chuẩn lớn nhất: 0.[23 số 1]\*2-127. Tuy nhiên IEEE754 quy định là 0.[23 số 1]\*2-126 vì muốn tiến gần hơn với số dương dạng chuẩn nhỏ nhất.

x 0000 0000 1111 1111 .... + Số dạng không chuẩn nhỏ nhất: 1.[22 số 0]1\*2-127. Tuy

nhiên IEEE754 quy định là 0.[22 số 0]1\*2-126.  $\times 0000 0000 0000...001 = 2^{-149}$ 

Trong C: phạm vi kiểu float từ 10-38 đến 1038, kiểu double từ 10-308 đến 10308. Precision: số bit được sử dụng để biểu diễn 1 giá trị. Accuracy: đô chính xác mà một kiểu biểu diễn có thể

hiểu diễn 1 giá tri. Rounding: phần cứng hỗ trơ 2 bit nhớ cho phần định trị giúp làm tròn kết quả.

#### Chuẩn IEEE làm tròn số chấm đông:

- + tròn lên: 1.01 10 -> 1.10, -1.01 10 -> -1.01
- + tròn xuống: ... -> 1.01, .... -> 1.10 + tròn về 0: bỏ giá tri 2 bit nhớ.
- + tròn về giá trị gần nhất.

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

## MACH Số

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

Mach số: thiết hị điện tử kết nối các linh kiên điện tử (transistor) hoạt động ở 2 mức điện áp (cao và thấp). Cổng logic: các linh kiên điện tử thường kết nối với nhau thành các khối cơ bản là cổng logic với giá tri luân lý 1 và 0 tương ứng với 2 mức điên áp là thế cao/thấp.

Tên cổng	Hình vẽ	Ký hiệu
AND	1	x.y
OR	1>-	x+y
XOR	<b>D</b> -	x⊕y
NOT	>-	$\bar{x}$ hoặc x'
NAND	$\Rightarrow$	$\overline{x}.\overline{y}$
NOR	-D-	$\overline{x+y}$
NXOR	<b>→</b>	$\overline{x \oplus y}$

Thiết kế logic: các cổng logic được kết nối với nhau thành các khối cao cấp hơn, gồm:

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

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

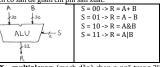
Các mach số (mach xử lý) được thiết kế ở mức logic sau đó dùng các kỹ thuật khác nhau để chuyển thành mạch số ở mức các linh kiên điên tử. Dùng để thiết kế counter, register, cache memory. Thiết kế mạch tổ hợp: lập bảng chân trị, xây dựng hàm

luân lý, vẽ sơ đồ, thử nghiêm. SOP: (sum of products): đầu ra 0 nhiều hơn 1.

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

Luât cơ bản: not(A+B) = not(A).not(B)

A.not(A) = 0 A + not(A) = 11 + A = 1 A.A = A not(A.B) = not(A) + not(B) Mach công: công bán phần gồm 2 ngõ vào 1 ngõ ra và cộng toàn phần gồm 3 ngõ vào 2 ngõ ra, không được vẽ dựa vào hàm bool mà được tạo nên bằng cách ghép 2 sơ đồ mạch công 2 bit -> ưu tiên tạo mạch bằng cách ghép mạch có sẵn để giảm chi phí sản xuất.



MUX - multiplexor: (mạch dồn) chọn n ngõ trong 2<sup>n</sup> ngõ vào để quyết đinh giá tri ngõ ra duy nhất. DEMIIX: (mach tách) chon n để lưa chon xem ngô vào

in sẽ được chuyển đến ngõ ra nào trong 2<sup>n</sup> 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 dãy bit nạp 10010000.

# KIÉN TRÚC MIPS

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

Bô lênh: (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 gơi nhớ ngôn ngữ máy -> hợp ngữ

Hợp ngữ: ngôn ngữ cấp thấp, cung cáp cách thể hiện gợi nhớ cho các lệnh máy. Dùng mã giả, tên gọi (label, tên biến,..) để ghi nhớ các mã lệnh, địa chỉ lưu 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 phu thuộc vào assembler mà tác giả sử dụng. Có thể tồn tai nhiều assembler cho cùng 1 assembly language.

ISA: (instructor set architecture) là tập lệnh dành cho những bộ vị xử lý khác nhau có kiến trúc tương tư 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 trong hệ thống nhúng), PowerPC của IBM.

RISC & CISC: complex là bô lênh gồm nhiều lênh 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 gon -> xử lý nhanh, tăng tốc đô xử lý cho những trường hợp thường xuyên xảy ra, thiết kế tốt đòi hòi sư 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.

opcode rs rt rd shamt funct 6 5 5 5 5 6					
opcode	shamt	funct			
6	5	5	5	5	6
oncode: mã	than tá	c cho b	iết loại	lênh ơì	

funct: kết hợp với opcode để xác định lệnh làm gì, trường này không nằm sát opcode vì nếu bỏ trường funct 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 lênh 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 5bit đủ để biểu diễn 32 reg

# Lệnh hợp ngữ số 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 đối xử 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 bô xử lý. Khuyết điểm là số lương cố đinh và han chế. Lưu biến: 8 thanh ghi thường sử dụng là \$16-\$23, đặt tên gơi nhớ là \$s0 - \$s7 (saved)

Cộng có dấu: add \$s0, \$s1, \$s2 (bù 2) Cộng không dấu: addu \$s0, \$s1, \$s2

Trừ có dấu: sub \$s0, \$s1, \$s2 (bù 2) Trừ không dấu: subu \$s0, \$s1, \$s2

MIPS cố đinh 32 bit và các lênh đơn giản để tăng tốc đô xử lý nên xây dựng lênh có nhiều toán hang nguồn sẽ tăng độ phức tạp câu lệnh -> 2 thanh ghi nguồn là đủ.

Lưu tam kết quả trung gian: 8 thanh ghi thường dùng là \$8 - \$15. đánh số \$t0 - \$t7. Thanh ghi zero: \$0 hay \$zero hỗ trơ phép gán và thao tác với 0. Vì sự đơn giản nên không cần thêm lệnh gán

giá tri 1 thanh ghi vào 1 thanh ghi, đồng thời lênh gán sẽ chỉ dùng 2 thanh ghi -> vị pham nguyên tắc.

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

Phép dịch: toán hạng thứ 2 phải là hằng số. sll \$s1, \$s2, 2 -> dịch trái luận lý \$s2 bit lưu vào \$s1,

tương tư 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 bô nhớ thì các toán hang giữ địa chỉ ô nhớ. Thao tác với bô nhớ cần ít nhất 1 toán hang nguồn là 1 toán hang đích. Cấu trúc I-format: cấu trúc lệnh tạo để thao tác với hộ

nhớ, giảm thiểu thay đổi so với R-format. Địa chỉ vùng nhớ được xác định bằng tổng của một thanh ghi chứa địa chỉ vùng nhớ (xem như con trỏ) và 1 số nguyên (xem như đô dời -byte).

opcode	rs	rt	immediate	
6	5	5	16	
ncode: tiror	g tir R-forma	t không có fi	inct	

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 -215 đến 215-1, đủ lớn để chứa giá trị độ dời (offset) từ địa chỉ trong thanh ghi cơ sở rs nhằm phụ vụ việc truy xuất hô nhớ trong lw và sw.

Data transfer instructions: lênh lưu trữ dữ liêu 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 trở lại bộ nhớ.

## Lệnh di chuyển dữ liệu:

opt opr, opr1(opr2) opt: tên thao tác, opr: thanh ghi lưu từ nhớ, opr 1: hằng số nguyên, opr2; thanh ghi chứa địa chỉ vùng nhớ

Nap: 1 từ dữ liêu bô nhớ vào thanh ghi lw \$t0, 12(\$s0) -> nap từ nhớ có địa chỉ (\$s0+12) vào thanh ghi \$t0.

Lưu: 1 từ dữ liêu thanh ghi vào bô nhớ

sw \$t0, 12(\$s0) -> lưu giá trị trong thanh ghi \$t0 vào vùng nhớ có địa chỉ (\$s0+12).

Nguyên tắc Alignment Restriction: các đối tương lưu trong bộ nhớ phải bắt đầu tại địa chỉ là bộ số của kích thước đối tương. Đối với MIPS, bôi 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: 12345678h

	12	34	56	78	
Địa chỉ:	0	1	2	3	(BIG)
	3	2	1	0	(LITTLE)

Con trỏ và giá trị: giử sử add \$t2, \$t1, \$t0 thì z = x+y nhưng lw \$t2,0(\$t0) thì \$t0 chứa địa chỉ tương đương con trỏ z=\*x.

Nan lini 1 hyte: loadbyte lb storebyte sh Cú pháp tương tư với lw, sw. Hỗ trơ các thao tác với ký

Vd: lb \$s0, 3(\$s1) -> nap giá tri byte nhớ địa chỉ (\$s1+3) vào byte thấp của \$s0, 24 bit còn lại có giá trị theo bit dấu của giá trị 1 byte. Nếu không muốn các bit còn lại có giá tri theo bit dấu dùng lênh lbu (unsigned).

Nap. lưu ½ từ nhớ: 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 đô bằng cách sử dụng các kỹ thuật song song. Trình biên dịch: ánh xạ các biến được sử dụng trong chương trình thành các thanh ghi (spilling).

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

Công hằng số có dấu: addi \$s0, \$s1, -10 Công hằng số kọ đấu: addiu \$50 \$51 10

Tính toán luân lý: andi, ori.

Vấn đề: giới hạn trường hằng số chỉ có 16 bit -> tăng kích thước lênh thao tác với hằng số có cấu trúc I-format Load Upeer Immediate: lui reg, imme, dwa hang 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 vào \$t0.

lui \$at. 0xABAB ori \$at, \$at, 0xCDCD

add \$t0, \$t0, \$at Tràn số: add/addi/sub 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). Rẽ nhánh có điều kiên: (I-format) cần 2 toán hang nguồn để so sánh và 1 toán hang cho biết địa chỉ cần beq register1, register2, L1 -> reg1=reg2

PC-Relative Addressing: immediate là số lênh cần nhảy qua để tới được nhãn, xem như số có dấu công 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 215 từ nhớ PC (217 bytes), đáp ứng hầu hết các yêu cầu nhảy lặp (tối đa 50) hoặc if (khoảng 3200).

bne register1, register2, L1 -> reg1!=reg2

Địa chỉ rẽ nhánh: nếu không thực hiện rẽ nhánh thì PC = PC+4 (địa chỉ lênh kế tiếp) nhưng nếu có thì PC = (PC+4) + imme\*4. -> lênh 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, chỉ

cần 1 toán hạng cho biết địa chỉ cần nhảy tới. opcode target address

Cấu trúc này có thể nhảy trong khoảng 226 từ nhớ (228 byte). Thông thường không cần thiết nhảy tới các từ nhớ có địa chỉ từ 227 - 232 nhưng nếu cần thiết thì có lênh ir. j label -> beq \$0, \$0, label

So sánh không bằng: slt reg1, reg2, reg3 Nghĩa là if (reg2 < reg3) reg1=1 else reg1=0 Căp slt->bne tương đương if (x<y) goto z.

Hằng số so sánh không bằng: (I-format) hữu ích với vòng lặp for: slti \$t0, \$s0, 1.

Lời goi thủ tuc: khi goi thủ tục, lênh tiếp theo được thực hiện là lênh đầu tiên của thủ tục -> xem tên thủ tục là một nhãn và lời gọi thủ tục là lênh nhảy tới nhãn này. jal label (jump and link, j-format) lưu địa chỉ của lệnh kế tiếp vào thanh ghi \$ra và nhảy tới nhãn label.

ir register (jump register) nhảy tới địa chỉ nằm trong thanh ghi register. Các thanh ghi mới: lưu trữ dữ liêu phục vụ cho thủ tục:

đối số (\$a0, \$a1, \$a2, \$a3), kết quả trả về (\$v0, \$v1), địa chỉ quay về (\$ra). Nếu sử dụng nhiều hơn -> dùng stack. Thủ tuc lồng nhau: địa chỉ trả về trong \$ra sẽ bị ghi đè bởi địa chỉ trả về của hàm khác được gọi lồng, nên cần lưu địa chỉ trả về của hàm bên ngoài trước khi gọi hàm bên trong.

Sử dụng stack : con trỏ ngăn xếp, thanh ghi \$sp được dùng để định vị vùng ngăn xếp, cần khai báo kích thước vùng ngăn xếp bằng cách tăng giá trị con trỏ ngăn xếp. Đầu thủ tục: entry label:

addi \$sp, \$sp, -framesize (1) sw \$ra, framesize-4(\$sp) (2)

(1): khai báo kích thước cần dùng cho stack. (2); cất địa chỉ trả về của thủ tục trong \$ra vào stack.

Cuối thủ tục: lw \$ra, framesize-4(\$sp) (3) addi \$sp, \$sp, framesize (4) ir \$ra

(3): khôi phục \$ra, lấy lại địa chỉ quay về của thủ tục đã lưu trong stack vào \$ra.

(4): kết thúc dùng ngăn xếp.

Sử dụng thủ tục: R(caller) gọi E (callee)

Trong R: (1) lưu địa chỉ trả về (trong \$ra) của R vào stack, (2) gán các đối số (nếu có) của R truyền vào E, (3) goi ial E

Trong E: (4) lưu vào stack các reg trong R có thể bi thay đổi trong E, ..., (6) khôi phục dữ liệu trong stack, (7) gọi ir \$ra để trở lại R.

Một số syscal	System	Arguments	Result
Service	Call Code	Arguments	Hesuit
print_int	1	\$a0 = integer	
print_float	2	\$£12 = float	
print_double	3	\$f12 = double	
print_string	4	\$a0 = string	
read_int	5		integer (in \$v0)
read_float	6		float (in \$£0)
read_double	7		double (in \$£0)
read_string	8	\$a0 = buffer, \$a1 = length	
sbrk	9	\$a0 = amount	address (in \$v0)
exit	10		
print_character	11	\$a0 = integer	
read character	12	Tara III	char (in \$v0)

Lệnh giả: các lệnh hợp ngữ không có cài đặt lênh máy tương ứng, nhùm giúp cho việc lập trình hợp ngữ dễ

IV.	IIPS INSTRUCTION						
R-Format	I-Format	J-Format					
add, sub, jr, and, or, nor sll, srl, sra, slt	addi, lw, sw, lh, sh, lb, sb, lui, andi, ori, beq, bne, slti	j, jal					
PSEUDO MIPS							
R-Format	move, mi	ult					

multi, li (load imme), blt (<), ble I-Format (<=), bgt (>), bge (>=)

Tóm tắt ý nghĩa thanh ghị:

\$zero	0	Chứa giá trị 0, ko đổi
\$at	1	Assembler temporary dành cho
		mục đích khác, hạn chế dùng
\$v0, \$v1	2, 3	Lưu giá trị trả về của hàm
\$a0 - \$a3	4-7	Lưu tham số truyền vào
\$t0 - \$t7	8-15	Lưu biến tạm
\$s0 - \$s7	16-23	Lưu biến
\$t8, \$t9	24, 25	Lưu biến tạm
\$k0 - \$k1	26, 27	Dùng cho nhân HĐH
\$gp	28	Pointer to global area
\$sp	29	Stack pointer
\$fp	30	Frame pointer
\$ra	31	Return address

Các thanh ghi không đổi: \$s0-\$s7 nếu callee thay đổi các thanh ghi này thì phải phục hồi lai giá tri trước khi kết thúc. Tương tự, giá trị \$sp ko đổi trước và sau khi gọi jal, nếu không thì caller sẽ không quay về được.

Các thanh ghi có thể thay đổi: gọi lệnh jal sẽ làm thay đổi \$ra -> lưu vào stack nếu cần. Tương tư \$a0-\$a3 và \$t0-\$t8, lưu lại giá trị nếu cần sau các lời gọi thủ tục;

#### THIẾT KẾ BÔ XỬ LÝ Lệnh máy: dãy bit chứa yêu cầu mà bộ xử lý phải thực

hiện, bao gồm mã thao tác (opcode-4bit ), toán hạng (operand-12bit) Hoạt động của CPU: xử lý lệnh máy qua 2 bước (chu kỳ

lệnh) gồm nạp lệnh (di chuyển lệnh từ bộ nhớ vào reg) và thực thi lệnh (giải mã, thực hiện thao tác yêu cầu). CPU: nhân lênh tại bộ nhớ PC -> MEM -> IR.: giải mã lênh IR -> CIJ -> giải mã -> phát tín hiệu: nhân dữ liệu tại bộ nhớ hoặc TBNV: địa chỉ -> ngăn nhớ -> tập thanh ghi; xử lý dữ liệu ALU -> tính toán -> thanh ghi dữ liệu;

ghi dữ liệu địa chỉ -> tân thanh ghi -> ngăn nhớ. CU: điều khiển nhân lênh tiếp theo từ bô nhớ, đưa vào IR rồi giải mã; tăng nôi dung của PC để trỏ vào lênh tiếp theo; phát ra các tín hiệu điều khiển thực hiện lênh; điều khiển các tín hiệu bên trong và bên ngoài CPU, điều khiển dữ liệu từ reg vào ALU và từ CPU ra reg, điều khiến bộ nhớ và modul vào ra.

Thanh ghi MAR (memory address): lưu đia chỉ được gửi ra/nhân vào từ bus địa chỉ.

Thanh ghi MBR (memory buffer): lưu giá trị được gửi ra/nhân vào từ bus dữ liêu. Bộ đếm chương trình (thanh ghi PC): lưu trữ địa chỉ của lênh tiếp theo được thực thị, quản lý địa chỉ vùng

lệnh, vùng dữ liệu được quản lý bởi thanh ghi con trỏ dữ liệu, được update trong chu kì nạp lệnh Thanh ghi IR(instruction register): lưu lênh được xử lý. Bô xử lý di chuyển lênh từ vùng nhớ có địa chỉ trong thanh ghi PC vào thanh ghi IR. Giá tri PC tăng 1 lương

bằng chiều dài của lênh. Chu kì lệnh: tính địa chỉ lệnh -> nạp lệnh -> giải mã lệnh -> tính địa chỉ toán hang -> nap toán hang -> thực thi

lệnh -> tính địa chỉ toán hạng chứa kết quả -> ghi kq. Quá trình thực thi: bô xử lý giải mã lênh trong IR, thực hiện các thao tác yêu cầu như tính toán, di chuyển data giữa thanh ghị, bộ nhớ/IO, thao tác rẽ nhánh,

Quá trình nap lênh: MAR <- (PC),

MBR <-MEMORY, IR <- (MBR), PC <- (PC)+1 CPU ra bus: điều khiển đọc/ghi ngăn nhớ, cổng vào ra và xử lý các tín hiệu từ bên ngoài gửi đến.

Bus đến CPU: tín hiệu xin ngắt, xin nhường bus.

MAR ontrol IR MBR

NN bậc cao(compiler)->Hợp ngữ (assembler)->mã máy. + Compiler phu thuộc vào ngôn ngữ bậc cao và kiến trúc hệ thống phần cứng bên dưới đang chạy. Compiler được biên dịch bằng assembler.

+ Một bộ vi xử lý có thể có nhiều assembler.

Linker: tập tin thực thi .exe, .bat, .sh,... Loader: chương trình tính toán và tải vào memory để CPU xử lý. Bô xử lý MIPS:9 lệnh add, sub, and, or lw, sw, beq, slt, j. CPU 1 chu kỳ: tất cả các công đoạn của 1 lệnh được xử

lý trong 1 chu kỳ đồng hồ (đủ lâu để xử lý mọi lênh).

Các bước thiết kế CPU:

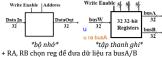
Bước 1: phân tích kiến trúc bộ lệnh ISA -> các vêu cầu về datapath:

- Trình bày từng lênh dưới dang register tranfers
- language (RTL) để thấy rõ ý nghĩa của từng lệnh. + nap lệnh: {op, rs, rt, imm16} <-MEM[PC]
- + add: R[rd] <- R[rs] + R[rt] PC<-PC+4
- Datapath phải có thành phần lưu trữ (main memory/cache) cho regs trong ISA, phải hỗ trơ thực thi tất cả các lệnh (cần memory để lưu lệnh/data regs để đoc rs/rt hoặc ghi rt/r, PC reg, sign extender).

Bước 2: lựa chọn các khối mạch cần thiết để xây dựng datapath (tổ hợp/tuần tư).

Khối mạch tổ hợp (adder/mux/alu); khối lưu trữ:

- + Bộ nhớ gồm đường dữ liệu vào data-in, đường dữ liệu ra data-out, đường địa chỉ để xác định từ nhớ được truy xuất, tín hiệu write-enable=1 xác định dữ liệu có được ghị vào hộ nhớ qua đường dữ liệu hay không.
- + Thanh ghi: (mach lât D) gồm N bit data-in và N bit data-out và tín hiệu write enable (0: dữ liệu trong thanh ghi không đổi. 1: ghi dữ liệu từ data-in vào thanh ghi) Tập 32 thanh ghi gồm 2 đường truyền dữ liệu ra là busA busB, một đường truyền dữ liệu vào busW.



- + RW chọn reg để ghi data từ busW khi write enable=1. Bước 3: lắp ráp các khối mạch đáp ứng yêu cầu của bô lệnh, gồm 5 công đoạn là sự tổ hợp đầy đủ của các thao tác cần thiết, thời gian mỗi công đoạn không quá chênh lệch nhau. (x86 nhiều công đoan hơn).
- (1) nạp lệnh: IR<- mem[PC] nạp lệnh 32bit từ bô nhớ tai đia chỉ trong PC vào thanh ghi lênh, sau đó PC=PC+4 để nạp lệnh kế tiếp.
- (2) giải mã lệnh: sẽ xác định được các giá trị RA RB RW tương ứng Rs Rt Rd và tín hiệu điều khiến RegWr.... phân tích các trường.
- (3) thực thi lênh: số học, luân lý, so sánh dùng 9 lênh mips trừ lệnh jump.
- (4): truy xuất hộ nhớ: chỉ có lw và sw do 2 việc này tốn nhiều thời gian nên cần công đoạn riêng.
- (5) ghi kết quả vào reg (add, sub, or, lw, slt)

Bước 4: phân tích mỗi lênh để xác định các tín hiệu điều khiển cần thiết.

Bước 5: thiết kế mạch cho các tín hiệu điều khiển

Vấn đề: xây dựng datapath phức tạp để xử lý lệnh sẽ khó khăn và không hiệu quả -> chia thành các công đoạn nhỏ (stages), xây dựng khối xử lý cho từng công đoạn rồi lắp thành datapath để dễ thiết kế và dễ thay đổi, tối ưu một công đoạn mà ít ảnh hưởng đến công đoạn khác.

Han chế của CPU 1 chu kỳ: không còn được sử dụng vì không hiệu quả + Tất cả công đoan của 1 lênh phải xử lý trong 1 chu kỳ

- theo tín hiệu đồng bộ nên các thành phần mạch có khả năng dùng chung đều được tách riêng (phần tính toán ALII/adder và phần lưu trữ instruction mem/data mem) -> sơ đồ phức tạp hơn.
- + Một chu kỳ phải đủ lâu để xử lý các lênh phức tạp, vd như lw phức tạp nhất (5 công đoạn) trong khi các lệnh khác chỉ mất 3 (beq) hoặc 4 (r-format) công đoạn. Chương trình có IC (instruction count) thì lênh xử lý trong 800xIC (ps =  $10^{-9}$ s)

CPU nhiều chu kỳ: mỗi công đoan lệnh thực hiện trong 1 chu kỳ (đủ lâu để thực hiện mọi công đoạn lệnh, time = 824xIC (ps)

+ Do mỗi công đoan được thực thi trong một chu kỳ riêng, nên có thể ghép các thành phần mạch dùng chung mà không xảy ra đung đô. (ALU+adder, IMem+Dmem) + Cần thêm các thanh ghi để lưu giữ kết quả trung gian của các công đoạn lệnh.

## KIẾN TRÚC x86 - 32 BIT

Intel 4004 - 4 bit (1971) là vi xử lý đầu tiên của Intel. Intel 8080 - 8 bit (1972): thanh ghi/đường truyền dữ liêu có 8 bit, đường truyền địa chỉ 16 bit (có thể truy xuất RAM 64KB), được sử dụng trên mấy tính cá nhân đầu tiên Altair.

Intel 8086/8088 - 16 bit (1978): thanh ghi 8 bit, đường truyền dữ liệu 16 bit (8088: 8 bit), đường truyền địa chỉ 20 bit, được dùng trên máy tính cá nhân IBM PC đầu tiên

Intel 80286 - 24 bit (1982): truy xuất bộ nhớ 16MB đường truyền địa chỉ 24 bit.

Intel 80386/i386 - (x86 32 bit/IA32) (1985): thanh ghi 32 bit, đường truyền địa chỉ 32 bit, có addressing modes mói.

Intel 80486/i486 - 32 bit (1989): kỹ thuật đường ống (pipelining).

Pentium - 64 bit (1993): đường truyền dữ liệu 64 bit, siêu vô hướng (2 đường ống ss). Athlon64 của AMD - (x86 64 bit) 2003: bộ vi xử lý

v86-64 hit đầu tiên

2004 - 2008: pentium 4 prescott, core 2, core i3, i5, i7, Antom,... core i9 2017.

Một vài đặc điểm: lệnh có độ dài 1-15 bytes, có 1 toán hạng vừa đóng vai trò toán hạng nguồn vừa đóng vai trò toán hang đích, có thể có một toán hang có thể truy xuất tới bộ nhớ, các chế độ định vị của các toán hang rất đa dang và phức tạp => dễ xây dựng các thành phần của mach xử lý để tăng tốc thời gian xử lý, bù lai complier phải có khả năng nhận diện lệnh không hiệu quả.

Chế đô thực: 16 bit dữ liêu, đường truyền 20 bit địa chỉ có thể truy xuất 1MB bô nhớ chính, MS-DOS.

Chế độ bảo vệ: 32 bit có thể truy xuất 4GB bộ nhớ chính, Windows, Linux. Chế đô 8086 ảo có chế đô thực dưới sự quản lý của 2 chế độ bảo vệ, cho phép hoạt động đồng thời ở 2 chế đô

Chế độ quản lý hệ thống: quản lý nguồn cung cấp, chẩn lỗi và bảo mật hệ thống.

Tổ chức bô nhớ ở chế đô thực: ở dang segments có size 64KB nằm chồng nhau mỗi segment cách nhau 16bytes.

Chuyển đổi địa chỉ logic -> vật lý ở chế độ thực: phy\_add = segment\*10h + offset. Vd: logic add = 1234h:0005h có phy add = 1234h\*10h

+0005h = 12345hVât lý -> logic: căp thanh ghi SS:ESP chứa logic add có

dang segment;offset. Do các đoan overlap nên mỗi ô nhớ có thể thuộc một segment khác nhau do đó ứng với 1 phy\_add có theer có nhiều logic\_add khác nhau.

7 mode địa chỉ: immediate, direct, indirect, register direct, register indirect, relative, indexed.

Tổ chức bộ nhớ ở chế độ bảo vệ: cũng chia thành các đoạn nhưng size không được định sẵn nên để định vị thị phải sử dung một bảng mô tả các đoan.

Thanh ghi trên x86 chứa địa đoạn (CS, DS, SS) và địa chỉ ô để truy xuất bô nhớ.

Chay chương trình trên hệ thống: thường chiếm 3 đoạn bộ nhớ: đoạn mã lệnh (code segment), đoạn dữ liêu (data segment) và đoan ngăn xếp (stack segment) dành để lưu các giá trị trung gian hoặc các địa chỉ trả về khi gọi hàm.

Một số thanh ghi khác: CS\_EIP là thanh ghi chứa địa chỉ lênh EIP 32bit kết hợp CS 16bit, thanh ghi trang thái chứa cờ điều khiển.

Thanh ghi cờ EFLAGS 32bit gồm 2 loại là cờ điều khiển và cờ phép toán; chứa kết quả của một lênh tính toán, giá trị được thiết lập sau khi mỗi lệnh được thực khi.

CF: carry cò tràn không dấu.

OF: overflow cò tràn có dấu.

SF: sign cờ dấu = 1 khi kết quả nhỏ hơn 0.

ZF: zero cò zero = 1 khi kết quả bằng 0.

AF: auxiliary cò nhớ (ra) từ bit 3 vào bit 4. PF: parity cò chẵn lẻ = 1 khi số bit 1 chẵn.

DF: direction cò cho biết chiều truy xuất 1 chuỗi, = 1 là truy xuất từ nhớ tới nhỏ.

Thanh ghi khác hỗ trơ HĐH: IDTR (16bit), GDTR (48bit). LDTR (48bit). TR (16bit)...

Cấu trúc lệnh x86: có tổng cộng 16byte nhưng thực tế chỉ cho phép dài nhất 15byte.



VD: định vị thanh ghi ADD EXB, EAX; định vị gián tiếp dựa trên thanh ghi cơ sở và độ dời ADD EBX, [ESI+45], đinh vi theo bôi (scaled indexed) ADD ECX, [EBX +

Cú pháp: 2 loai chính là Intel và AT&T <le><lenh><toán\_hang\_đích><toán\_hang\_nguồn> Các toán hang: thanh ghi EAX, AX; ô nhớ [EBX],

[EBX+ESI+7], biến;hằng số. Lênh nhảy: thực hiện dựa vào giá trị của các trường cờ S, Z, C, O,... thường dùng lệnh cmp trước các lệnh nhảy để thay đổi giá trị các cờ.

Ngăn xếp: LIFO dùng theo chiều giảm của địa chỉ (khác với vùng nhớ thông thường).

PUSH<toán hang> toán hang có thể là thanh ghi/vùng nhớ/hằng số 4byte, làm giảm ESP đi 4 sau đó đưa giá tri

POP<toán hang> toán hang có thể là thanh ghi/vùng

nhớ 4byte, đưa giá trị từ ô nhớ có địa chỉ SS:ESP vào toán hang sau đó tặng ESP lên 4. CALL<tên thủ tục> dùng stack để push địa chỉ lệnh tiếp

ngay sau lệnh CALL (nơi cần quay lại), sau đó ghi vào thanh ghi con trỏ lênh EIP địa chỉ của lênh đầu tiên của chirong trình con

<tên thủ tuc>PROC: khai báo thủ tuc.

RET: lệnh này pop giá trị từ đỉnh ngăn xếp vào thanh ghi con trỏ lênh EIP, làm cho lênh tiếp theo được thực hiện chính là lênh ngay sau lênh CALL.

MOV AX, BX: copy giá trị của BX vào AX.

của toán hạng vào ô nhớ có địa chỉ SS:ESP.

LEA: load effective address.

Tính toán số học, luân lý: add, sub, inc, dec, imul, idiv (by EDX:EAX), CMP (dùng thêm flag reg), and, or, xor, shl, shr,...

Control flow: jg, ja, jc,..., loop,...

Một số lưu ý: khai báo thủ tục sau lời gọi thủ tục thoát, không có lênh nhảy ra ngoài thủ tục, lời gọi thủ tục thao tác với ngăn xếp một cách không tường minh, nếu thân thủ tục có thao tác với ngăn xếp nhớ lưu lai địa chỉ trả về của thủ tục truyền tham số cho thủ tục (thanh ghi ngăn xếp, biến toàn cuc).

Kỹ thuật pipeline: các lệnh được thực thi chồng lên nhau. Tăng tốc độ xử lý bằng cách tăng throughput (khối lương công việc có thể hoàn thành trong một khoảng thời gian) của tất cả công việc chứ không giảm latency (thời gian hoàn thành) của một công việc.

Program execution T	ime -	20	00	40	00	64	00	8	00	10	00	12	00	1400
order (in instruction											The	yi g	i an	xử lý:
lw \$1,100	(\$0)	Instruction fetch		Reg	AL	U	Da		Reg	-	7 ×	20	0 =	1400 (ps
lw \$2, 200	(\$0)	200 ps	Instru	action ch		Reg	AL	.U	Dat		Rog			
lw \$3, 300	(\$0)		200	) ps	Instru fet			Reg	ALI	U	Dat		Reg	
, 						ps			200					ps

Tốc độ: non-pipeline (ICx5)x200 (ps) còn đối với pipeline (4+IC)x200 (ps). Trường hợp lý tưởng (thời gian của các công đoạn bằng nhau) thì thời gian giữa 2 lênh liên tiến nineline hằng thời gian giữa 2 lênh liên tiến không pipeline chia cho số công đoan. (xét trong ví du thì bằna 160ps = 800/5)

Sư tăng tốc: trường hợp IC lớn trong điều kiến lý tưởng thì pipeline sẽ nhanh gấp N lần so với non-pipeline với N là số công đoạn. Nhưng trong thực tế các công đoạn không bằng nhau nên áp dung pipeline phải chon công đoan dài nhất để làm một chu kì pipeline nên mức tăng số sẽ nhỏ hơn số công đoạn.

Vấn đề: xảy ra các xung đột nghĩa là lệnh tiếp theo không thể thực thi trong chu kỳ pipeline ngay sau đó (hoặc sẽ có nhưng cho kết quả sai), thường do 1 trong 3 lý do:

Xung đột cấu trúc (structural hazard): xảy ra do hai lênh cùng truy xuất vào một tài nguyên phần cứng nào đó cùng lúc nên phần cứng không thể hỗ trơ (ALU, mem, reg) -> Ghi vào thanh ghi nửa chu kỳ đầu, đọc thanh ghi nửa chu kỳ cuối; tách ALU và adder; tách Imem và Dmem (cache)

Xung đột dữ liệu (data hazard): lệnh sau có thể phụ thuộc vào dữ liêu của lệnh trước nên có thể xảy ra trường hợp dữ liệu mà lệnh này cần vẫn chưa sẵn sàng -> kỹ thuật nhìn trước (forwarding/bypassing), Stall, Non (~Stall)

Xung đột điều khiển (control/branch hazard): lênh nạp vào không phải là lệnh được cần, xảy ra khi luồng thực thi chứa lênh nhảy. Khi thực hiên rẽ nhánh sẽ không biết lệnh tiếp theo sẽ được thực hiện là lệnh nào nên overlap không hiệu quả -> kỹ thuật Stall, Delayed Branch, Branch Prediction

## SO SÁNH CÁC KIẾN TRÚC

MIPS: kiến trúc 3 toán hạng (2 nguồn, 1 đích), ít lệnh nên tốc đô xử lý nhanh hơn. Kiến trúc nap-lưu chỉ có lệnh load/store để truy xuất bộ nhớ, các lệnh còn lại thao tác trên thanh ghi/hằng số, các lênh có chiều dài cố định mạch xử lý đơn giản hơn nên dễ nâng cao tốc đô bằng cách sử dụng các kỹ thuật song song hóa.

x86 32bit: kiến trúc 2 toán hang (1 toán hang nguồn và 1 toán hang đóng 2 vai trò), lệnh ngắn hơn, ít lệnh hơn nên mã nguồn nhỏ hơn. Kiến trúc thanh ghi-bô nhớ. tất cả các lệnh đều có thể truy xuất bộ nhớ. Lệnh có kích thước thay đổi nên kích thước mã nguồn có thể nhỏ hơn, sử dụng cache hiệu quả hơn, hằng số có thể 8 hoặc 32 hit

#### HÊ THỐNG LƯU TRỮ Về chức năng: thanh ghị, bộ nhớ trong, bộ nhớ ngoài.

Theo vật lý: bộ nhớ bán dẫn, bộ nhớ từ, bộ nhớ quang; có thể khả biến hoặc không, có thể xóa được.

Thanh ghi Cache ape. CD. DVD, BD, USB, me

Thanh ghi: cấu tao từ mạch lật, nằm trong CPU, có dụng lương nhỏ nhất nhưng tốc đô truy xuất nhanh nhất, được dùng trong các bộ xử lý (lưu lệnh và dữ liệu). Có thể làm bằng nhiều công nghệ khác nhau: trigger, core, thin film. Thường tổ chức thành tập register file. Chứa các thông tin tạm thời, ít nhất 3 loại thanh ghi địa chỉ.

ROM: là bô nhớ không khả biến, chỉ đọc không thể ghi; được chế tạo từ transistors, diode PROM: programable ROM, ghi được 1 lần duy nhất.

EPROM: eraseable PROM, có thể viết nhưng phải xóa bằng tia cực tím sau khi ghi.

EEPROM: electrically EPROM, xóa bằng điện, ghi/xóa tirng hyte

FlashROM: không thể xóa từng byte mà phải xóa từng khối. Tốc độ ghi/xóa cao -> người dùng dễ dàng upgrade dung lương

RAM: là hô nhớ khả biến, đơn vị lưu trữ là điện tích trong tụ điện, không phải hiệu điện thế nên giá thành thấp nhưng tốc độ truy xuất chậm, tăng khoảng 7% mỗi năm. Gồm RAM tĩnh được chế tạo từ mạch lật và không cần làm tươi (SRAM) và RAM đông được chế tao từ tu điện (DRAM).

SDRAM: (synchronous RAM) viêc truy xuất được đồng bô bởi tín hiệu đồng hồ bên ngoài.

Bộ nhớ chính (main mem/RAM) được làm từ công nghệ SDRAM, do bộ xử lý đánh địa chỉ trực tiếp +Thế hệ 1: SDR SDRAM (single data rate) 1 chu kỳ đồng

hồ chỉ chuyển được 1 lần 64bit dữ liệu. +Thế hê 2: DDR SDRAM (double data rate) 1 chu ký

đồng hồ chuyển được 2 lần dữ liêu -> có tốc đô gấp đôi SDR (lý thuyết), chuyển được 64bit dữ liêu. DDRII gấp 2 lần DDRI., DDRIII gấp 34lần DDRI. Bộ nhớ Cache: là mức thanh ghi gần nhất, dùng công

nghệ SRAM, có tốc độ cao hơn bộ nhớ chính. Đóng vai trò làn bộ nhớ đệm truy xuất nhanh giữa CPU và main mem. Lưu trữ tam thời bản sao một phần nội dung main mem nhằm giảm truy xuất main mem; hoạt động bằng nguyên lý định vị tham số bộ nhớ, truyền dữ liệu giữa CPU và cache theo đơn vị từ nhớ, giữa cache và bộ nhớ chính theo đơn vị block/line nhớ.

Nguyên lý hoạt động cache: cục bộ về thời gian (temporal locality - nếu một ô nhớ được dùng đến trong thời điểm hiện tại thì nó dễ có khả năng được dùng đến lần nữa trong tương lai gần) và cục bô về không gian (spatial locality - nếu một ô nhớ được dùng đến trong thười điểm hiện tại thì những ô lân cân có khả năng sắp được dùng đến).

Khi CPU/IO đọc 1 ô nhờ từ main mem: kiểm tra xem đã có trong cache chưa, nếu có (cache hit) thì đọc nôi dung trong cache không cần truy xuất main mem, nếu chưa (cache miss) thì chép khối nhớ chứa ô nhớ cần đọc từ main mem vào cache rồi vào CPU. Thời gian xử lý cache miss là miss nenalty

Thiết kế để hiệu quả: cách tổ chức, chiến lược thay thế, đồng bộ hóa nội dung trong cache và main mem, kích thước của cache, kích thước một phần tử của cache và số lương/loại cache.

Tổ chức bộ nhớ cache: phần tử gọi là line gồm data block (4bytes) và tag chứa địa chỉ của block.

Truy nhập: liên kết đối với cache, ngẫu nhiên với bộ nhớ trong và trực tiếp với đĩa từ.

Fully Associate Mapping: một block có thể được nạp vào bất kỳ line nào. Do 1 block có thể được nạp vào bất cứ line nào nên để xác đinh block đã nằm trong line hay chưa thì nhải duyệt tất cả các line -> chi nhí tìm kiếm cao, dùng nhiều bit cho địa chỉ tag

Block/Tag s Word w Vd: 24bit address, 222 blocks, block size = 4 bytes => w =

Direct Mapping: cải thiên nhược điểm của FAM bằng cách sử dụng hash table. Mỗi block Bị được ánh xa dụy nhất vào một line Li với Li = Bj mod L với L là số line. Đơn giản, nhanh, chi phí tổ chức thấp nhưng không uyển chuyển (nếu truy xuất tuần tư các địa chỉ cùng 1 line thì liên tục cache miss dù cache còn trống nhiều chỗ; mỗi block cố định 1 cache line).

Tag r-s Cache line r

Word w

Vd: 24bit address, 222 blocks, 214 lines, block size = 4 bytes => w = 2 r = 14 % tag = 8

K-way Set Associate Mapping: kết hợp ý tưởng giữa direct mapping và associate mapping. Các line được chia làm S tập hợp bằng nhau, mỗi tập có K phần tử, Si = Bi mod S, các line được quản lý theo kiểu fully

rag s-u	Cache set <b>u</b>	woru <b>w</b>				
Vd: 24bit addre.	ss, 2 <sup>22</sup> blocks, 2 <sup>14</sup> lines,	block size = 4bytes				
=> w = 2, tag = 9, line = 13.						
Chiến lược th	av thể: không có lư	a chon cho direct				

mapping. Với FAM và SAM: có thể chon random, FIFO, LRU (least recently used- dùng nhiều), LFU (least frequently used). Đồng bô hóa cache và main mem: writethrough xảy

ra khi line/block bị thay đổi trong cache/main mem bởi CPU/IO thì block/line tương ứng trong main mem/cache sẽ lập tức được cập nhật. Tuy nhiên nếu có quá nhiều thay đổi -> châm. Hoặc writeback: khi line thay đổi trong cache sẽ dùng bịt đánh dấu, khi phải thay thể line bị đánh dấu thì cập nhật block tương ứng trong main mem, IO truy xuất main mem qua cache. Kích thước block: nếu quá nhỏ sẽ giảm tính lân cân về

không gian, nếu quá lớn thì thời gian chuyển block vào cache lâu (miss penalty lớn), thường từ 8-64bytes. Kích thước cache: kích thước càng lớn giá thành càng

cao. Kích thước nhỏ thì số lương block ít dẫn đến cache miss cao còn quá lớn thì mất nhiều thời gian kiểm tra tồn tại của block do có nhiều nội dung không cần thiết. Tùy vào cách tổ chức để đánh giá tốc đô.

Số lương và loại: dựa vào vị trí thì có thể chia thành on-chip cache (mức thấp, nằm cùng IC với bộ xử lý) và off-chip cache (mức cao, nằm trên IC riêng và được nối với bộ xử lý thông qua Back Side Bus BSB). Thường sử dụng 2 loại là: unified cache (dùng chung

cho cả lệnh và dữ liệu) và split cache (dùng riêng cho lênh và dữ liêu). Theo level có L1 (dùng riêng) và L2, L3 (dùng chung).

Thời gian truy xuất trung bình:

AMAT = L1 hit-time + L1 miss-rate\*L1 miss-penalty. Nếu có dùng L2 thì L1 miss-penalty = L2 hit-time + L2 miss-rate \* L2 miss-penalty -> dùng nhiều cache level thì tốc đô truy xuất cao hơn.

Xây code hợp lý: những trường hợp thường xảy ra có tốc độ nhanh (vòng lặp), minimize lần cache miss trong mỗi loop.

Đĩa từ: thiết bị lưu trữ dữ liêu lâu dài phổ biến nhất, gồm đĩa mềm (châm, 1 lớp đĩa) và đĩa cứng (nhanh hơn, nhiều lớp đĩa). Gồm nhiều đĩa tròn (platter), mỗi lớp phủ 1 hoặc 2 mặt, mỗi mặt có 1 đầu đọc (head) và nhiều đường tròn đồng tâm (track), mỗi đường tròn có nhiều cung tròn (sector) chứa khoảng 4096 điểm từ = 512byte -> mỗi lần đọc ghi khoảng 512byte.

## Cơ chế đọc dữ liệu: thời gian truy xuất đĩa từ:

disk latency = seek time + rotation time + transfer time. Seek time: thời gian di chuyển đầu đọc tới đúng track cần đọc, phu thuộc vào số track/mặt và tốc độ actuator. Rotation time: thời gian đĩa quay sao cho sector muốn đọc nằm dưới đầu đọc.

Transfer time: thời gian đọc và truyền dữ liệu, phụ thuộc vào độ phủ từ và chuẩn giao tiếp.

Dung lương đĩa từ: head/disk \* track/head \* sector/track \*0.5KB.

Kỹ thuật RAID (redundant array of inexpensive disks) kết hợp nhiều ổ đĩa (vật lý) thành một hệ thống đĩa logic duy nhất bằng phần cứng (RAID controller) hoặc phần mềm => đảm bảo an toàn dữ liêu, tăng tốc đô truy xuất hệ thống. Dữ liệu lưu trên đĩa sẽ được lưu đồng thời trên tất cả các đĩa. Một số RAID thông dụng 0, 1 10, 5, 6. Cơ chế kiểm tra và sửa lỗi: lỗi xuất hiên do các xung điện hoặc điện từ trường. Phương pháp là sử dụng mã kiểm tra bằng cách thêm các bịt vào để phát hiện lỗi (parity bit, CRC, ECC, hamming code,...)

Đĩa quang: đĩa CD (compact dics) gồm 4 lớp; đĩa DVD (digital versatile disc), đĩa HD DVD & Blue-ray dics -> không còn nhổ hiến do sự thay thế của flash memory Flash memory: công nghệ lưu trữ được sử dụng phổ biến nhất hiện nay: USB, memory card, ROM. Không cần nguồn điên để duy trì dữ liêu. Có thể duy trì dữ liêu lâu dài, ít tốn điện năng. Nhưng số chu kỳ ghi/xóa bi giới hạn (do sự hao mòn), hầu hết khoảng trên 100 ngàn hoặc trên 1 triệu chu kỳ ghi/xóa dữ liệu.

Tape library: thiết bị cho phép kết hợp hàng ngàn băng từ để tạo thành một thiết bị lưu trữ có dụng lượng lên đến terabyte, petabyte.

HÊ THỐNG NHẬP XUẤT

Disk library: kết hợp đĩa cứng. Optical jukebox: kết hợp đĩa quang. Vai trò: giao tiếp giữa thành phần xử lý của máy tính với các đối tương bên ngoài giúp các đối tương này có thể cung cấp yêu cầu và dữ liệu cho thành phần xử lý. Tiêu chí phân loai: trên chức năng, đối tương tương

tác hoặc tốc đô nhập xuất. Tổ chức kết nối các thiết bị nhập xuất: do sư chênh lệch tốc độ giữa các thiết bị nhập xuất, với tốc độ truy xuất CPU nên thường dùng nhiều cấp độ bus khác nhau

để kết nối các nhóm thiết bị nhân xuất. Các loại bus liên kết hệ thống: điều khiển, dữ liêu, đia

Truy xuất thiết bị nhập xuất:

+ port-mapped: mỗi thiết bị được gán một hoặc vài port. Truy xuất vào các port này tương ứng truy xuất thiết bị. Trong bộ lệnh của bộ xử lý thường phải có một số lệnh chuyên biệt để thao tác với thiết bị nhập xuất như IN và OHT của x86

+ memory-mapped: mỗi thiết bị được ánh xạ vào một hoặc vài vùng nhớ. Truy xuất vào những vùng nhớ tương ứng với truy xuất vào thiết bị, ví dụ như lw và sw của MIPS.

## 3 cơ chế giao tiếp:

+ polling (còn gọi là programmed I/O): CPU điều khiển toàn bô quá trình nhập xuất, chủ đông kiểm tra tình trạng thiết bị, sau đó gửi yêu cầu truy xuất và truyền data. CPU phải chờ trong suốt thời gian truy xuất thiết

+ interrupt-driven: CPU không phải chờ trong suốt quá trình truy xuất thiết bị, thiết bị chủ đông báo tình trang với CPII tốn chi nhí xử lý ngắt + direct memory access DMA: khi truyền dữ liêu lớn,

phải ngắt CPU nhiều lần (trên mỗi đơn vị truyền dữ liêu). Hê thống hỗ trơ DMA controller để giao tiếp với các thiết bi cần truy xuất khối lương lớn dữ liêu, chỉ ngắt CPU một lần cho một yêu cầu truy xuất dữ liệu.