TỔNG QUAN VỀ MÁY TÍNH & BÔ XỬ LÝ - Các thế hê:

+ 0 (1642-1940) - máy tính cơ học - Non-digital computers

+ 1 (1940 -1956) - bóng đèn chân không Vacuum Tubes (xe như các bit), má y tính đầu tiên IBM 700 (dòng thinkpad hiện nay), hê thống ENIAC (Electronic Numerical Intergrator and Computer), tốc đô 40.000 ops/s, (Ex: UNIVAC, EDVAC, EDSAC) + 2 (1956-1963) - transistor IBM 7094 tốc độ x5 là hước độ phá, ra đời ngôn ngữ Fortran, ngôn ngữ Assembly (FORTRAN, COROL) +3(1964 - 1971) - Integrated Circuits(Ex: PDP8, IBM 360, ICL

2900) (mạch tích hợp/ bán dẫn, 3000 devices/chip, tốc độ x5. + 4 (1971 - nav) mạch tích hơn mật độ siêu cao LSI/ vị xử lý. xử lí đầu tiên ra đời(intel 4004), GUI cũm dc phát triển trong gó này (Ex: IBM, STAR 1000), hơn 100tr devices/chip, tốc độ x10. + 5 (Turong lai - 2010 - nay): sự ra đời của AI, Parallel Processin Kiến trúc Von-Neumann: dữ liệu và chương trình chứa trong bi đọc ghi, bộ nhớ được đánh địa chỉ cho các ngặn nhớ không phi 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 lương transistor (tích hợp trong một IC/chip hoặc trê mỗi đơn vị inch2) tăng gấp đôi mỗi 18 hoặc 24 tháng. Thành phần cơ bản của máy tính (để bàn): màn hình, bo mạc

chủ (mainboard), CPU, chân cắm dây nối HDD, RAM, chân cắm mỏ rộng PCI/PCI Express, nguồn điện, Ő quang CD/ DVD, ổ đĩa cứng bàn phím, chuột, Bộ xử lý: xử lý các lệnh máy, gồm khối điều khiển (điều khiến xử

lý ALU và data trên register), ALU, register, (internal bus - kết nổ CU, ALU và register). 2 thành phần bên trong vi xử lý: Control Unit, Data Path

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

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

Chin: là mạch tích hơn 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 vị xử lý đồ họa), RAM (bộ nhớ truy cập tức thời phục vụ CPU), bán cầu bắc (tích hợp trên mainboard hỗ trợ truyền thông tin cho CPU/RAM, nằm kế CPU), bán cầu nam (quản lý thiết bi ngoai vi như HDD, mousse, keyboard, nằm cuối mainboard).

Chức năng cơ bản của máy tính: lưu trữ + xử lý + trao đổi dũ 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: khả năng tạm ngừng chương trình để thực thi chương trìn

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 vi 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 (sự cố điện/ lỗi bộ nhớ), mức độ ưu tiên cao nhất: ngắt mềm do lệnh INT (ROM-BIOS): ngắt ngoại lệ do lênh của CPII như chia 0 flag Các thành nhần của hệ thống máy tính:

Đơn vị xử lý trung tâm (Central Processor Unit - CPU)

Bộ nhớ chính: RAM & ROM Hệ thống vào ra (Input-Outpu System)

Liên kết hệ thống (Buses): bus địa chỉ, bus dữ liệu hoặc bus cu

Các loại máy tính:

Super Computer: xử lí lượng dữ liệu lớn Work Station:hệ thống một người sử dụng

Microcomputer:cho cá nhân sử dụng

△Cấu tạo: Processing devices: mother board, processer, RAM ROM, switched mode power supply.

Phần cứng	Phần mềm
4 loại: thiết bị vào/ra,	2 loại: phầnmềm ứng
Secondarystorage device,	dụng,phần mềm hệ
Internalcomponent	thống
Chương 2: BIẾU	DIÊN SỐ NGUYÊN
rr 2 2 / 1 1/2 1/3	

Không dấu: $0 \rightarrow 2^n - 1$

Dấu lượng: $-(2^{n\cdot 1}-1)\rightarrow 2^{n\cdot 1}-1$, n bit biểu diễn $-(2^{n\cdot 1}-1)$ đến $2^{n\cdot 1}$ -1, phức tạp cho máy tính phân biệt số 0. -127 đến +127 **Bù 1:** $-(2^{n-1}-1) \rightarrow 2^{n-1}-1$, giới hạn, hạn chế giống lượng dấu. Bù 2: $-2^{n-1} \rightarrow 2^{n-1} - 1$, giới hạn $-2^{n-1} \rightarrow 2^{n-1} - 1$, khắc phục vấn đề co

2 số 0 của lượng dấu và bù 1. -128 đến +127 Số quá: $0 - K \rightarrow 2^{n-1} - K - 1$, n bit thì $k = 2^{n-1} - 1$, N > k thì +, < thì -

♦ x SHL v = x . 2v ♦ x SHR v = x / 2v

- AND dùng để tắt bit (AND với 0 luôn = 0)
- OR dùng để bật bit (OR với 1 luôn = 1) XOR, NOT dùng để đảo bit (XOR với 1 = đảo bit đó)
- ♦ x AND 0 = 0
- \bullet v XOR v = 0
- Mở rộng:
- Lấy giá trị tại bit thứ i của x: (x SHR i) AND 1
- Gán giá trị 1 tại bit thứ i của x: (1 SHL i) OR x
- Gán giá trị 0 tại bịt thứ i của x: NOT(1 SHL i) AND x
- Đảo bit thứ i của x: (1 SHL i) XOR x Vấn đề tràn số: x + y < -2w-1 negative overflow

x + v > 2w-1 - 1: positive overflow

Phép toán công/trừ : bù 1 nếu dư bịt MSB thì công tiếp, bù 2 do thì bỏ.

Nhân biết tràn số: với số không dấu thì nhớ 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 công hai số cùng dấu cho ra

2. 4. 8. 16. 32. 64. 128. 256. 512. 1024. 2048. 4096. 8192. 16384. $32768,65536 (2^1 \rightarrow 2^16)$

Thuật toán nhân không đấu: C (1bit) = 0 và A = 0. Khởi tao đãy 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 dãy CAO, Res = AO.

Thuật toán Booth - nhân số bù 2: A = 0. 0.1 (1 bit) = 0. Khởi ta $d\tilde{a}v$ AOO1M. Län: xét OoO-1 nếu = 01 thì A=A+M, nếu = 10 th A=A-M, sau đó dịch phải dãy AOO-1. Kết quả = AO. Thuật toán chia số ko dấu: A = 0. O là số hị chia M là số chia Khởi tạo dấy AQ. Lăp: dich trái AQ, A = A-M nếu A >0 thì Qo=1, ngược lại Qo=0 $var{A} = A+M$. Res = 0. du=A.

M = 7, Q = -3, n = 4

| | | Α | Q | Q. ₁ | M |
|----------|----------|---------|------|-----------------|------|
| Khởi đầu | ı | 0000 | 1101 | 0 | 0111 |
| Bước 0: | A=A-M | 1001 | 1101 | 0 | 0111 |
| | shift | 1100 | 1110 | 1 | 0111 |
| Bước 1: | A=A+M | 0011 | 1110 | 1 | 0111 |
| | shift | 0001 | 1111 | 0 | 0111 |
| Bước 2: | A=A-M | 1010 | 1111 | 0 | 0111 |
| | shift | 1101 | 0111 | 1 | 0111 |
| Bước 3: | shift | 1110 | 1011 | 1 | 0111 |
| Kết quả | 11101011 | l = -21 | | | |

Ví du phép chia

| A | Q | M = 001 |
|------|---------------|---------------|
| 0000 | 0111 | Initial value |
| 0000 | 1110 | Shift |
| 1101 | 1110 | Subtract |
| 0000 | 1110 | Restore |
| 0001 | 1100 | Shift |
| 1110 | 1100 | Subtract |
| 0001 | 1100 | Restore |
| 0011 | 1000 | Shift |
| 0000 | | Subtract |
| 0000 | 1001 | Set $Q_0 = 1$ |
| 0001 | 0010 | Shift |
| 1110 | | Subtract |
| 0001 | 0010 | Restore |
| | (a) $(7)/(3)$ | |
| | | |

Kết quả: Q là thương, A là số du

Thuật toán

chia số bù 2:

thực hiện như

nếu số bị chia

và số chia khác

dấu thì đổi dấu

Khởi tạo: A = r

{Chuyển sang

trái (SHL) [A,

Nếu A < 0: 0

= 0 và A +M→A

Ngược lại

phép chia

không dấn

kết quả.

hit $0 \cdot k = n$

A-MA

k=k-1}

International Electrotechnical Commission

| Name | Abbr | Factor |
|------|------|---|
| kibi | KI | 210 = 1,024 |
| mebi | Mi | 2 ²⁰ = 1,048,576 |
| gibi | Gi | 230 = 1,073,741,824 |
| tebi | Ti | 2 ⁴⁰ = 1,099,511,627,776 |
| pebi | Pi | 2 ⁵⁰ = 1,125,899,906,842,624 |
| exbi | Ei | 260 = 1,152,921,504,606,846,976 |
| zebi | Zi | 270 = 1,180,591,620,717,411,303,424 |
| yobi | YI | 280 = 1,208,925,819,614,629,174,706,176 |

| Name | Abbr | Factor | SI size |
|-------|------|---|--|
| Kilo | K | 215 = 1,024 | 10 ³ = 1,000 |
| Mega | M | 2 ²⁰ = 1,048,576 | 10 ^E = 1,000,000 |
| Gga | G | 2 ³⁰ = 1,073,741,824 | 10 ² ±1,000,000,000 |
| Tera | T | 24 = 1,099,511,627,776 | 1012 a 1,000,000,000,000 |
| Peta | P | 2 ⁵⁰ = 1,125,899,905,842,624 | 1015 = 1,000,000,000,000,000 |
| EAN | E | 210 = 1,152,921,504,606,846,976 | 1018 = 1,000,000,000,000,000,000 |
| Zetta | Z | 27 = 1,180,591,620,717,411,303,424 | 10 ²¹ = 1,000,000,000,000,000,000,000 |
| Yotta | Y | 281 = 1,208,925,819,614,629,174,706,176 | 1074 x 1,000,000,000,000,000,000,000,000 |
| | | | |

- Chú ý: khi nói "kilobyte" chúng ta nghĩ là 1024 byte nhưng thực ra nó là 1000 bytes theo chuẩn SI, 1024 bytes là kibibyte (IEC)
- Hiện nay chỉ có các nhà sản xuất đĩa cứng và viễn thông mới dùng n 30 GR → 30 * 109 ~ 28 * 230 hytes
- n 1 Mbit/s → 106 b/s

CHƯƠNG 3: BIỂU DIỄN SỐ THỰC

Giới hạn số chấm tĩnh: n bit thì phần thập phân nhỏ nhấ 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.

 $X = (-1)^S \cdot 1, M \cdot R^(E-127)$ Double: độ dài 64bit với các trường 1+11+52.

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

 $X = (-1)^S \cdot 1.M \cdot R^{(E-16383)}$ Số chấm đông: 1 bit dấu (sign), 8 bit mũ (E) và 23 bit đin

 $tri(S) \rightarrow giátri = S \times 2^{E}$ Các số đặc biệt: số 0 (mũ - Exponent = 0, trị - Significan = 0), Số không thể chuẩn hóa (denormalized) (mũ = 0, tri #0). Số vô cùng (infinity) (mũ = 111...1 (toàn bit 1), tri = 0), Số báo lỗi (NaN - Not a Number) (111...1 (toàn bit

Pham vi biểu diễn:

1), tri #0)

 Dang chuẩn lớn nhất: 1.[23 số 1]*2127. x 0 | 1111 1110 | 1111 1111 = (2-2-23)*2¹²⁷.

tương tự cho số âm nhỏ nhất -(2-2-23)*2127) + Dạng chuẩn nhỏ nhất: 1.[23 số 0]*2-126.

x 0000 0001 0000 0000 = 2-126

Cấu tao của 1 lênh máy (Instruction): Là dãy bit chứa yêu cầu mà bộ xử lý trong CPU (ALU) phải thực hiện. Gồm 2 thành phần Số dang không chuẩn lớn nhất: 0.[23 số 1]*2-127. Tuy nhiên IEEE754 guy định là 0,[23 số 1]*2-126 vì muốn tiến chính: Mã lệnh (opcode): thao tác cần thực hiện gần hơn với số dương dạng chuẩn nhỏ nhất

Hardware.

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

nhiên IEEE754 quy định là 0.[22 số 0]1*2-126. x 0000 0000 0000...001 = 2-149

Trong C: pham vi kiểu float từ 10-38 đến 1038, kiểu double

từ 10-³⁰⁸ đến 10³⁰⁸. Precision: số bịt được sử dụng để biểu diễn 1 giá tri. Accuracy: độ chính xác mà một kiểu biểu diễn có thể biểu

diễn 1 giá tri. Rounding: phần cứng hỗ trợ 2 bịt nhớ cho 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 bịt 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 ký tư kẻ

khung. Biểu diễn số thực sau theo dạng số chấm động chính xác don (32 bit): X = -3050 Bước 1: Đổi X sang hệ nhị phân

C = -305010 = -1011 1110 1010

Bước 2: Chuẩn hóa theo dạng ±1.F * 2E K = -305010 = - 1011 1110 1010 = -1.01111101010 * 2^11 ◆

Bước 3: Biểu diễn Floating Point Số âm: bit dấu Sign = 1

Số mũ E = 11 → Phần mũ exponent với số thừa K=127 được biểu → Exponent = E + 127 = 11 + 127 = 13810 = 1000 1010 / Phần

định trị = 0111 1101 0100 0000 0000 000 (Thêm 12 số 0 cho đủ → Kết quả nhận được: 1 1000 1010 0111 1101 0100 0000 0000

Chương 4: Kiến trúc bộ lênh Các thành phần của bô vi xử lý: 3

-Control unit : - Dùng để ra các tín hiệu điều khiến cho ALU, giả mã các câu lênh.

-ALU (Đơn vị toán học luận lí) : Chiu trách nhiệm trong các nhiệm vụ thực hiện các câu lệnh, phép tính. Đọc các toán hạng từ Tập thanh ghi, Sau đó hoặc là lưu lại vào Tập thanh ghi , hoặc à ghi vào trong bộ nhớ.

-Registers (Tập thanh ghi) Cách hoat đồng của bộ vị xử lý



Instructions(lênh/chỉ thi/ mã máy) Là 1 cái chuỗi bit chứa /êu cầu gửi đến CPU (ALU) thực hiện

Gồm 2 thành phần chính

Opcode (Mã lệnh) : Cho ALU biết thao tác cần thực hiện Operand (Toán hạng) : Cho biết ALU biết toán hạng các đô ượng bị tác động bởi thao tác chứa trong mã lệnh

*Có 2 trường phái thiết kế bộ lệnh; Complete Instruction Se Computer(CISC); bộ lệnh gồm rất nhiều lệnh, từ đơn giản đến phức tạp Reduced Instruction Set Computer(RISC); bộ lệnh chỉ ồm các lệnh đơn giản.

Cisc sẽ nhấn manh trên Hardware hơn. Risc thường sử dụn cho các hệ thống nhúng như MIPS Để chay những tập tin mã máy cần Linker& Loader

KL: Code => Compiler => Hop ngữ

Assembler => Object file, Linker liên kết những cái Object file đó thành file Thực thi .exe ; File .exe đó sẽ được Loader đưa vào Bô nhớ để cho CPU đọc instruction để thực hiện.

Phân biệt 3 khái niệm:

Ngôn ngữ lập trình (High-level language): Loại ngôn ngữ nhân tao, được cấu thành bởi từ vựng và ngữ pháp. Giúp người lập trình diễn đạt hướng dẫn cho máy tính hoạt đông. Có đô trừu tượng cao, con người dễ hiểu nhưng máy tính kho hiểu (cần phiên dịch).

Ngôn ngữ máy (Machine language): Là đãy bit (0/1) chứa yêt cầu mà bộ vi xử lý (CPU) phải thực hiện. Ngôn ngữ riêng của CPU, khó hiểu với con người nhưng máy tính hiểu trực tiếp. Hợp ngữ (Assembly language): Ngôn ngữ cấp thấp, rất gần với ngôn ngữ máy. Dùng ký hiệu mã giả (mnemonic) thay cho dãy bit, dùng **tên (label, tên biến, tên chương trình)** thay cho địa chỉ. Để lập trình hơn ngôn ngữ máy, đủ để con người hiểu và sử dụng tốt hơn. Phu thuộc vào kiến trúc bộ lệnh

Compiler và Assembler:

Compiler: Trình biên dịch ngôn ngữ cấp cao sang hợp ngữ. Phụ thuộc vào ngôn ngữ cấp cao được biên dịch và kiến trúc hệ thống phần cứng bên dưới mà nó đang chay. Assembler: Trình biên dịch hợp ngữ sang ngôn ngữ máy. Phu thuộc vào kiến trúc bộ lệnh (ISA) và hệ điều hành

(OS). Một kiến trúc ISA có thể có nhiều Assembler khác nhau.

Thông tin về toán hạng (operand): các đối tượng bị tác đông bởi thao tác chứa trong mã lênh Kiến trúc bô lênh (ISA): Tân lênh dành cho những hộ vị xử lý có kiến trúc tương ti nhau. Trong cùng một dòng vi xử lý, bộ lệnh gần giống nhau.

Một số kiến trúc bộ lệnh thông dụng: 80x86 (gọi tắt x86) của Intel (IA-16, IA-32, IA-64), MIPS (dùng nhiều trong hệ thống nhúng), PowerPC của IBM. Trường phái thiết kế bô lênh CISC và RISC:

Hai trường phái thiết kế bộ lệnh:

Complete Instruction Set Computer (CISC): Bộ lệnh gồm rất nhiều lệnh, từ đơn giản đến phức tạp. Nhấn mạnh trên

Reduced Instruction Set Computer (RISC): Bô lênh chỉ gồn các lệnh đơn giản. Thường sử dụng cho các hệ thống nhúng như MIPS.

Bộ lệnh đại diện: x86 đại diện cho CISC, MIPS và PowerPC đạ diên cho RISC. Ưu nhược điểm (cơ bản): CISC có lệnh phức tạp hơn, có thể

hực hiện tác vụ trong ít lệnh hơn nhưng việc giải mã và thực thi có thể châm hơn và phức tạp hơn; RISC có lệnh đơn giản, dễ dàng thiết kế mạch xử lý pipeline tốc độ cao, nhưng cần nhiều lệnh hơn để thực hiện cùng tác vụ. Linker và Loader:

Linker: Chương trình dùng để liên kết các file object (đã biệ dịch từ mã nguồn) và các thư viên (Program library) có sắn lại với nhau để tạo thành tập tin thực thị (.exe. .bat. .sh...) Loader: Chương trình dùng để tính toán và tải tập tin thực

thi (từ đĩa cứng) vào bô nhớ chính (RAM) để CPU có thể xử lý. Quá trình tao file thực thi (.exe): Source file(s) (ngôn ngữ cấp cao) -> Compiler -> Assembly language program -> Assembler -> Object file(s) (mã máy). Các Object file và Program library -> Linker -> Executable

Quá trình xử lý lênh của CPU (Instruction Cycle): CPU xử lý lênh qua một vòng lặp gồm 2 bước chính: Nạp lệnh (Fetch cycle): Di chuyển lệnh từ bộ nhớ vào thanh ghi (IR) trong CPU. Các bước cơ bản: MAR ← PC, MBR ←

Memory[MAR], IR ← MBR, và cập nhật PC ← PC + Chiều dài Thực thi lênh (Execute cycle): Giải mã lênh (trong IR) và thực hiện thao tác yêu cầu (ví dụ: tính toán bởi ALU, truy cập bộ nhớ, điều khiến luồng chương trình). Gồm các bước như

tính địa chỉ toán hạng, nạp toán hạng, thực hiện phép toán, ghi kết quả Một số thanh ghi cơ bản trong CPU: PC (Program Counter): Luru địa chỉ của lệnh kế tiếp trong b

nhớ sẽ được nan IR (Instruction Register): Luru lênh hiện tại đạng được CPU giải mã và thực thị

MAR (Memory Address Register): Lưu địa chỉ của ô nhớ hoặc thiết bị I/O mà CPU đang truy cập. MBR (Memory Buffer Register): Luru dữ liêu được đọc t

hoặc ghi vào bộ nhớ/thiết bị I/O. Thanh ghi đa dụng (General Purpose Registers): Dùng

để lưu trữ tạm thời dữ liệu (toán hạng, kết quả trung gian) cho các phép tính của ALU. (Ví dụ: \$s registers cho biến lưu, \$t registers cho biến tam trong MIPS; AX, BX, CX, DX... trong x86). Thanh ghi cờ (Flag Register): Chứa các bit trạng thái về kết quả của phép tính hoặc hoạt động của CPU (ví dụ: cờ Zero, cò

Dấu, cờ Tràn). Thanh ghi ngăn xếp (Stack Pointer - SP): Lưu địa ch đỉnh của ngăn xếp (stack).

Thanh ghi địa chỉ nên (Base Pointer - BP): Lưu địa cl nền của khung ngăn xếp hiện tại

Thanh ghi địa chỉ trả về (Return Address - \$ra): Lưu địa chỉ của lệnh kế tiếp sau khi một lời gọi thủ tục (hàm) kết thúc.

CHƯƠNG 5: BỘ LỆNH LEGV8

LEGv8 là tập hợp con của kiến trúc tập lệnh ARMv8, phục giáo dục.

ARMv8 là kiến trúc tập lệnh 64-bit cho thiết bị hiện đại. LEGv8 đơn giản hóa ARMv8, giữ lại phần quan trọng để dễ học Mã nguồn mở

Đặc điểm:

Dưa trên **kiến trúc RISC**, tập lệnh nhỏ gọn, dễ hiểu. Kiến trúc 64-hit (hỗ trợ cả 32-hit nhưng tận trung 64-hit).

Bô lênh gồm: **Tính toán số học, Lệnh Logic, Lệnh di chuyển** dữ liệu. Lênh điều khiển rẽ nhánh

Tâp thanh ghi (Register File): Thanh ghị tổng quát

 32 thanh ghi kích thước 64 bit (DoubleWord): X0-X31. 32 thanh ghi con kích thước 32 bit (Word): W0-W31. Thanh ghi cờ (Flag Register - NZCV):

Bộ flags gồm Negative (N), Zero (Z), Carry (C) Overflow (V).

Fhanh ghi số thực (Floating Point Registers): 32 bit: S0-S31

64 bit: D0-D31 Chức năng các thanh ghi tổng quát cụ thể: X0-X7: Làm đối số / kết quả trả về của hàm X8: Chứa (vị trí) địa chỉ kết quả trả về.

X9 - X15: Thanh ghi tạm (Temporary registers). X16 - X17 (IPO - IP1): Thanh ghi tạm của linker hoặc tạn cho trường hợp khác (First/Second intra-procedure-call scratch registers).

X18: Thanh ghi nền tảng cho code không phụ thuộc nền MemtoReg = 0, ALUOp = 10, MemWrite = 0, ALUSrc = 0, RegWrite tảng hoặc làm thanh ghi tạm (Platform register).

X19 - X27: Thanh ghi lưu trữ (Callee-saved registers) X28 (SP): Trỏ đến đỉnh ngăn xếp (Stack Pointer).

X29 (FP): Trỏ đến khung trang (Frame Pointer) X30 (LR): Thanh ghi liên kết (Link Register - địa chỉ quay

XZR (X31): Chứa hằng số 0. fâp lênh (Instruction Set):

Cú pháp chung: opt opr, opr1, opr2 (số học/logic) hoặc opt onr. [onr1. #onr2] (load/store). Lênh tính toán số học:

Công/Trừ: ADD. SUB.

Cộng/Trừ hằng số: ADDI, SUBI.

Lệnh tác động cờ: ADDS, SUBS, ADDIS, SUBIS. Phén nhân: MUL, SMULH (nhân 64 bit cao), SMULL (nhâi

Phép chia: SDIV (chia có dấu), UDIV (chia không dấu). ênh Logic:

Bitwise AND, OR, XOR: AND, ORR, EOR.

Bitwise AND, OR, XOR hằng số: ANDI, ORRI, EORI.

Lênh tác đông cờ: ANDS, ANDIS. Phép NOT: Không hỗ trợ trực tiếp, thực hiện bằng EOR

ênh dịch luân lý: Dich trái: LSL (Logical Shift Left) - Thêm bit 0 bên phải. Dich phải: RSL (Right Shift Logical) - Thêm bit 0 bên trái ênh dịch số học:

Dịch trái: Không có lệnh dịch trái số học (giống LSL). Dich phải: RSA (Right Shift Arithmetic) - Thêm hit dấu hên trái

ênh di chuyển dữ liệu (Data Transfer): Load/Store DoubleWord: LDUR (Load Register), STUR

(Store Register) Load/Store Word (4 byte): LDURSW (Load Signed Word), STURW (Store Word). LDURSW nap 1 word (4 Đặc điểm byte) vào 4 byte thấp của thanh ghi 64 bit và mở rộng dấu.

Load/Store Halfword (2 byte): LDURH (Load Halfword), STURH (Store Halfword). LDURH nap 1 halfword (2 byte) vào 2 byte thấp và mở rông dấu. Load/Store Byte (1 byte): LDURB (Load Byte), STURB (Store Byte). LDURB nap 1 byte vào 1 byte thấp và mở Ưu điểm

Load/Store độc quyền: LDXR (Load exclusive register),

STXR (Store exclusive register) - Dùng trong trường hợp

nhiều luồng/tiến trình truy cập cùng vùng nhớ. Gán giá tri vào thanh ghi (Move Wide): Gán địa chỉ biến vào thanh ghi: ADRP (đưa địa chỉ cơ

sở), kết hợp với ADD và :lo12: (công offset). Lênh rễ nhánh (Branch Instructions):

Re nhánh không điều kiện (Unconditional Branch) Triển khai vòng lặp, if-else: if-else: Biến dịch thành các lệnh so sánh (CMP) rẽ nhánh có

tiều kiện (B.Cond) và rẽ nhánh không điều kiện (B) để bỏ qua động trên thanh ghi. (Ví dụ: x86: ADD EAX, [EBX]; LEGV8: cần 2 các khối code Vòng lặp (do-while, while, for): Biến dịch thành các lệnh so sánh (CMP) và rẽ nhánh có điều kiên (B.Cond hoặc CBZ/CBNZ) để quay lại điểm bắt đầu vòng lặp. Nguyên tắc chung là **viết lại**

dưới dạng goto và sử dụng lệnh rẽ nhánh có điều kiện. Thao tác với Stack: - Stack: Cấu trúc dữ liệu LIFO, sử dụng ngăn xếp (Stack) trong

ô nhớ chính Được định vị và quản lý bởi thanh ghi con trỏ ngăn xết (SP/X28).

Dùng để lưu tam các giá trị trong thanh ghị, địa chỉ trả về, biếi cuc bâ Đẩy (Push): Giảm SP, sau đó dùng STUR hoặc STR để lưu dữ

iêu vào địa chỉ mới của SP. (Ví du: SUB SP, SP, #8; STR X0, [SP]) Lấy (Pop): Dùng LDUR hoặc LTR để lấy dữ liệu từ địa chỉ của SP. sau đó tăng SP. (Ví du: LTR XO. [SP]: ADD SP. SP. #8)

Thủ tục / Hàm (Procedures): Thanh ghi liên quan:

Đối số (Argument input): X0 - X7 Kết quả trả về (Return value): X0 - X1 Thanh ghi không bảo toàn (Caller-saved): X0-X18 (Nếu

hàm được gọi thay đổi các thanh ghi này, hàm gọi (caller) nhải lưu chúng trước khi gọi). Thanh ghi bảo toàn (Callee-saved): X19-X28 (Nếu bàn

được gọi (callee) thay đổi các thanh ghi này, nó phải lưu chúng ở đầu hàm và khôi phục ở cuối hàm trước khi return).

Địa chỉ quay về (Link Return): X30 (LR). Gọi hàm: Dùng lệnh BL Ten_Ham.

Quay vê: Dùng lênh RET (hoặc BR LR), lấy địa chỉ từ X30 để hảy về

Thủ tục lồng nhau: Cần sử dụng Stack để lưu địa chỉ quay về của hàm ngoài (từ X30) trước khi gọi hàm lồng bên trong (vì hàm lồng sẽ ghi đè X30).

Format lệnh LEGv8: Tất cả lênh có kích thước cố đinh 32 bit.

Biểu diễn Little Endian.

-Các trường lệnh: opcode, Rm, shamt, Rn, Rd DT_address, ALU immediate. COND_BR_address, MOV_immediate. Các định dạng chính: R-format, D-format, I-format, B

format, CB-format, IW-format. Các định dạng khác nhau để giảm độ phức tạp của giải mã nhưng vẫn giữ kích thước lệnh đồng nhất 32 bit. 9.1. ADD X9. X10. X11

Reg2Loc = 0, UncondBranch = 0, Branch = 0, MemRead = 0,

9.2. LDUR X9, [X10, #0]

Reg2Loc = x, UncondBranch = 0, Branch = 0, MemRead = 1, MemtoReg = 1, ALIJOn = 00, MemWrite = 0, ALIJSrc = 1, RegWrit

9.3. STUR X9. [X10. #0] Reg2Loc = 1 UncondBranch = 0 Branch = 0 MemRead = 0 MemtoReg = v ALHOn = 00 MemWrite = 1 ALHSrc = 1 RegWrite

= 09.4. B I.1 Reg2Loc = x. UncondBranch = 1. Branch = 0. MemRead = 0.

MemtoReg = x. ALUOn = xx. MemWrite = 0. ALUSrc = x. RegWrite

(Cần thêm logic kiểm tra X9 ≠ 0, mở rộng đường so sánh) Reg2Loc = x, UncondBranch = 0, Branch = 1, MemRead = 0,

9 7 ADDI X9 X10 #1

Bao gồm các khối chính: Control Unit Registers ALU Data Memory. Instruction Memory. Có các tín hiệu điều khiển từ

Control Unit tới các khối khác (ví dụ: Reg2Loc MemWrite

dàng thực hiện trên phần cứng

- Kiến trúc **CISC (Complex Instruction Set Computer)

Thường có **một toán hạng vừa là nguồn vừa là đích**

Tập thanh ghi:

Thanh ghi doan (Segment Registers - 16 bit):

- Thanh ghi con trỏ/chỉ mục:

+ SP/ESP/RSP (Stack Pointer) - Tro dinh stack. + BP/EBP/RBP (Base Pointer) - Tro nen khung stack.

+ DI/EDI/RDI (Destination Index) - Chỉ số đích trong lênh

+ SI và DI có thể được sử dụng như thanh ghi đa dụng (c

- Thanh ghi cờ (EFLAGS - 32 bit): + Chứa **kết quả của phép tính** (cờ trạng thái): CF

chuỗi). IF (Interrupt Enable Flag - cho phép ngắt), TF (Trap Flag

- Thanh ghi lênh: IP/EIP/RIP (Instruction Pointer).

Các đoan **chồng nhau**, mỗi đoan cách nhau 16 bytes.

- Lệnh di chuyển dữ liệu:

là một kiến trúc RISC (Reduce

bản và ít biến thể. Hỗ trợ các chế đô vừ lý khác nhau nh

hypervisor mode.

Instruction Set Computer). Thiết kế đơn giản với một số lệnh co

Hiệu suất cao với các lệnh đơn giản, d

9.5. EOR X9, X10, X11 (Chỉ cần mở rộng ALU hỗ trợ EOR, datapath không đổi) Reg2Loc = 0, UncondBranch = 0, Branch = 0, MemRead = 0 MemtoReg = 0, ALUOp = 10, MemWrite = 0, ALUSrc = 0, RegWrite

9.6. CBNZ X9, L1

MemtoReg = x, ALUOp = 01, MemWrite = 0, ALUSrc = 0, RegWrite

(Không cần thêm datapath, chỉ cần ALUOp = cộng) Reg2Loc = x, UncondBranch = 0, Branch = 0, MemRead = 0. MemtoReg = 0, ALIJOn = 00, MemWrite = 0, ALIJSrc = 1, RegWrite

ALUSrc, RegWrite,...) để điều khiển hoạt đông của datapath

Là một kiến trúc CISC (Complex

Instruction Set Computer). Hỗ trợ một tập lệnh phức tạp và đa

dạng. Có nhiều biến thế và chế độ hoạt.

Hỗ trợ các tính năng phong phú

| | lệnh tí:
LEGv8 | hác biệt chính về truy cập
nh toán/logic thao tác trực
(và các kiến trúc Load-Stoi
nh Load/Store truy cập bộ | e ti | iếp trên ô nhớ, trong khi
khác như MIPS) chỉ cho |
|---|-------------------|--|------|--|
| , | Nhược
điểm | Phức tạp trong việc thiết kế và triển
khai trên phần cứng. Tổn nhiều
nguồn tài nguyên hơn so với các
kiến trúc RISC. | | Yêu cầu nhiều lệnh hơn để thực hiện
các tác vụ phức tạp so với CISC.
Cần phải có sự hỗ trợ phần cứng tốt để
tối ưu hiệu suất. |
| , | | Multiple Data) va cac lệnh phức tạp. Được sử dụng rộng rãi trong nhiều
ứng dụng máy tính và server. | | Tiết kiệm năng lượng và không gian bị
nhớ. |

lênh: LDUR X9. [X1.#offset]. ADD X0. X0. X9). CHIFONG 6: X86

Lênh có **đô dài thay đổi** (từ 1 đến 15 byte)

 Có thể có **một toán hạng từ bộ nhớ**. - Chế độ định vị toán hạng đa dạng và phức tạp.

- Thanh ghi đa dụng: + AX, BX, CX, DX (16 bit)

+ EAX, EBX, ECX, EDX (32 bit) + RAX, RBX, ... (64 bit - Y86-64, ngoài phạm vi 32bit chính).

+ CS (Code Segment) - Kết hợp với IP/EIP/RIP để xác định địa chỉ lênh.

+ SS (Stack Segment) - Kết hợp với SP/ESP/RSP hoặc

(Carry Flag), OF (Overflow Flag), SF (Sign Flag), ZF (Zero Flag) AF (Auxiliary Carry Flag), PF (Parity Flag).

chế đô chay từng lênh).

* Cấu trúc bộ nhớ (chế độ thực 8088/8086):

Địa chỉ vật lý = Segment * 16 + Offset.

BP/EBP/RBP cho stack.

+ DS (Data Segment) - Kết hợp với BX, DI, SI cho dữ liệu, + ES (Extra Segment) - Kết hợp với DI cho dữ liệu chuỗi

+ SI/ESI/RSI (Source Index) - Chỉ số nguồn trong lệnh

+ 80386 trở lên 32 bit: EIP, EBP, ESP, EDI, ESI (thanh ghi 32

+ Chứa **cờ điều khiển**: DF (Direction Flag - chiều xử lý

Tổ chức theo **đoan (segments)** có kích thước 64KB.

+ **MOV**: Dùng để chuyển giữa các thanh ghi, giữa thanh ghi và ô nhớ, giữa thanh ghi và hằng số. **Không tác động đến cờ**. Có thể chuyển giữa **thanh ghi đa dụng, thanh ghi đoan ô nhớ, hằng số**. **Không chuyển trực tiếp giữa 2 ô nhớ* hoặc giữa 2 thanh ghi đoạn

+ **XCHG**: Hoán đối nội dung giữa 2 toán hạng (thanh ghi/ô nhớ). **Không tác động đến cờ**. Toán hạng không được là thanh ghi đoạn

+ **PUSH**: Cất 1 từ (16 bit) hoặc 1 double word (32 bit) từ thanh ghi/ô nhớ vào đỉnh ngăn xếp. 'SP = SP - size'. **Không tác đông đến cờ**

+ **POP**: Lấy 1 từ/double word vào thanh ghi/ô nhớ tù đỉnh ngăn xếp. 'size => {SP}', 'SP = SP + size'. **Không tác động

+ **PUSHF/POPF**: Cất/Lấy nội dung thanh ghi cờ (Flags

vào/từ stack. + **IN/OUT**: Đọc/Ghi dữ liệu từ/ra cổng I/O. Thườ

dùng thanh ghi AL/AX. **Không tác đông đến cờ** + **Lệnh di chuyển chuỗi (MOVSB, MOVSW, MOVSD)*

Chuyển 1 byte/word/dword từ chuỗi nguồn (DS:SI) sang chuỗi đích (ES:DI). SI/DI tự động tăng/giảm (+/- 1, 2, 4 tùy DF). **Không tác đông đến cờ**. Lênh tính toán số học + logic:

+ **ADD**: Cộng. Thay đổi cờ: AF, CF, OF, PF, SF, ZF. + **ADC**: Cộng có nhớ (cộng với CF). Thay đổi cờ: AF, CF,

OF, PF, SF, ZF, + **SUB**: Trù. Thay đổi cò: AF, CF, OF. PF. SF. ZF.

+ **INC**: Công 1. Thay đổi cờ: AF, OF, PF, SF, ZF.

+ **DEC**: Trừ 1. Thay đổi cờ: AF OF PE SF ZF

+ **MIII.**: Nhân không dấu. Kết quả lưu ở AX (8hit nguồn hoặc DX:AX (16bit nguồn). Thay đổi cờ: CF, OF (được đặt nếu kết quả tràn). + **IMUL**: Nhân có dấu.

và AH/DX (du). + **IDIV**. Chia có dấu + **AND, OR, XOR**: Logic bit. Thay đổi cò: PF, SF, ZF và

xóa cờ CF, OF (=0). + **NOT **: Đảo từng bit.

+ **NEG**: Xác định số bù 2

+ **Lệnh dịch bit (Shift):**

* **SHL**: Dich trái logic (tương tự SAL).

* **SHR**: Dịch phải logic (thêm 0 bên trái). * **SAL**: Dịch trái số học (tương tự SHL).

* ** SAR**: Dich phải số học (thêm bit dấu bên trái).

* Chú ý: Toán hang không được là thanh ghi đoan tro lênh dịch/quay

+ **Lênh quay bit (Rotate):**

* **RCL**: Quay trái thông qua cờ nhớ (CF).

* ***RCR**: Quay phải thông qua cờ nhớ (CF).

* **ROL**: Quay trái. * **ROR**: Ouav phải.

+ **Lênh so sánh:**

CMP. So sánh 2 hute/từ cân nhật cờ (AF CF OF PI

* **CMPSB/CMPSW**: So sánh từng phần tử chuỗi

nguồn (DS:SI) và chuỗi đích (ES:DI), cập nhật cờ, SI/DI tự động tăng/giảm.

| X4X | | | | | | | | |
|--------|-----|-----|-----|-----|-----|------|-----|-----|
| X1X2X3 | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 000 | 0 | 1 | 3 | 2 | 6 | 7 | 5 | 4 |
| 001 | 8 | 9 | 11 | 10 | 14 | 15 | 13 | 12 |
| 011 | 24 | 25 | 27 | 26 | 30 | 31 | 29 | 28 |
| 010 | 16 | 17 | 19 | 18 | 22 | 23 | 21 | 20 |
| 110 | 48 | 49 | 51 | 50 | 54 | 55 | 53 | 52 |
| 111 | 56 | 57 | 59 | 58 | 62 | D 63 | 61 | 60 |
| 101 | 40 | 41 | 43 | 42 | 46 | 47 | 45 | 44 |
| 100 | 32 | 33 | 35 | 34 | 38 | 39 | 37 | 36 |

- Lênh lặp (LOOP):

+ LOOP, LOOPE/LOOPZ, LOOPNE/LOOPNZ. + Là lênh phối hợp giữa **DEC CX** và **INZ** (hoặc

+ LOOP: DEC CX, nhảy nếu CX != 0.

+ LOOPE/LOOPZ: DEC CX, nhảy nếu CX != 0 VÀ ZF = 1.

+ LOOPNE/LOOPNZ: DEC CX, nhảy nếu CX != 0 VÀ ZF = 0

Lênh gọi chương trình con (CALL):

+ 2 loai: CALL gần và CALL xa (tương tư IMP) + Dùng để **đẩy địa chỉ quay về lên stack**, sau đó nhá

+ CALL gần: đẩy IP lên stack, nhảy IP = địa chỉ thủ tục.

+ CALL xa: đẩy CS và IP lên stack, nhảy IP/CS = địa chỉ th

Lệnh phục vụ ngắt (INT và IRET):

+ **INT Number**: Goi chương trình con phục vụ ngắt (CTCPVN) số Number

+ Thực hiện INT: **Cất thanh ghi cờ. CS. IP vào stack*

sau đó nhậy tới địa chỉ CTCPVN lấy từ hẳng vector ngắt

+ **IRET**: Lênh trả về từ CTCPVN. **Lấy IP. CS. than ghi cờ từ stack** để quay về chương trình bị ngắt.

Triển khai if-else, switch-case:

 Sử dung lênh so sánh (CMP) và các lênh rẽ nhánh có điều kiện (JE, JNE, etc.) hoặc không điều kiện (JMP).

* Thao tác với stack (đã mô tả trong lệnh PUSH/POP).

Thao tác với mảng 1 chiều (đã mô tả trong chế đô địa chỉ).

* Thủ tục / Hàm:

- Sử dụng lệnh CALL để gọi và RET để quay về.

- Tham số: Thường truyền qua **thanh ghi ** hoặc **stack**

- Kết quả trả về: Thường đặt trong thanh ghi AX/FAX/RAX**

- Thanh ghi **caller-saved/callee-saved**: Turong to LEGv8, nhưng cu thể các thanh ghi thuộc nhóm nào cần tham khảo quy ước gọi hàm (calling convention). Thường AX, CX, DX là caller-saved (thủ tục gọi phải lưu); BX, BP, SI, DI là callee-save thủ tục được gọi phải lưu nếu dùng). CHUONG 7: MACH LOGIC

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

Mach số: thiết bi điên tử kết nối các linh kiên điên tủ (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 trị luận lý 1 và tương ứng với 2 mức điện áp là thế cao/thấp.

| | Ten cong | n ve | niệu | | |
|--|----------|----------|-----------------------------|--|--|
| | AND | | x.y | | |
| | OR | 1 | x+y | | |
| | XOR | 1>- | x⊕y | | |
| | NOT | | \bar{x} hoặc x' | | |
| | NAND | | $\overline{x}.\overline{y}$ | | |
| | NOR | → | $\overline{x+y}$ | | |
| | NXOR | → | x⊕y | | |
| Thiết kế logic: các cổng logic được kết nối với nhau thành | | | | | |

+ **DIV**: Chia không dấu. Kết quả lưu ở AL/AX (thương) các khối cao cấp hơn, gồm: Mach tổ hợp: kết quả của mạch chỉ phụ thuộc vào giá tri

đầu vào tại thời điểm đạng xét; adder, decoder, multiplexor, ALU,...Gồm n ngõ vào, m ngõ ra (hàm luân lý của các ngõ vào), propagation delay là độ trễ giữa thời điểm tín hiệu vào ổn đinh và ra ổn đinh. Mach tuần tư: kết quả của mạch không chỉ phụ thuộc vào

giá trị đầu vào mà còn nhụ 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 điện 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,

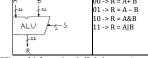
SOP: (sum of products): đầu ra 0 nhiều hơn 1. POS: (products of sum): đầu ra 1 nhiều hơn 0.

Luât cơ bản: not(A+B) = not(A).not(B) $A \operatorname{not}(A) = 0$

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

not(A.B) = not(A)1 + A = 1Mach công: công bán phần gồm 2 ngõ vào 1 ngõ ra và công

toàn phần gồm 3 ngõ vào 2 ngõ ra, không được vẽ dưa vào hàm bool mà được tạo nên bằng cách ghép 2 sợ đồ mạch cộng 2 bit -> ưu tiên tạo mạch bằng cách ghép mạch có sẵn để giảm chi phí sản xuất. 00 -> R = A+ B



MUX - multiplexor: (mạch dồn) chọn n ngõ trong 2ⁿ ngô vào để quyết định giá trị ngõ ra duy nhất.

DEMUX: (mach tách) chon n để lưa chon xem ngõ vào in sẽ được chuyển đến ngõ ra nào trong 2n ngõ ra.

ALU: được thiết kế đơn giản, cho phép lựa chọn và thực niên các phép toán ADD, SUB, AND OR. Mach lật: flin-flon là mạch tuần tự cơ bản nhất có chức

năng lưu trữ 1 bịt nhớ. Thanh ghi dịch: (shift register) 4 bit tạo từ 4 mạch lật R

với dãy bit nap 10010000.CHƯƠNG 9: HỆ THỐNG NHẬP XUẤT Chức năng: **Trao đổi thông tin** giữa thành phần xử lý của

máy tính (CPU, bô nhớ) với **thế giới bên ngoài** (các thiết bị ngoại vị như bàn phím, chuột, màn hình, ổ đĩa, mạng...). * Vấn đề: Tốc độ xử lý của CPU và bộ nhớ **nhanh hơn nhiều so với tốc độ hoạt động của các thiết bị ngoại vi đa dạng. Cần co

cơ chế để **cầu nối** sư chênh lệch tốc đô này. * Tổ chức kết nối thiết bi: - Sử dụng nhiều cấp độ **bus** khác nhau (bus điều khiển, us dữ liêu, bus địa chỉ) để kết nối các nhóm thiết bị có tốc đô

 Ví dụ sơ đồ hệ thống máy tính: CPU -> Chipset (Northbridge eu an lý bộ nhớ tốc độ cao, Southbridge - quản lý I/Ô tốc độ thấp)

PROM: programable ROM, ghi được 1 lần duy nhất rung bình, LPC cho thiết bị châm) -> Các I/O controller -> Thi oi ngoai vi.

* Hai cách tổ chức địa chỉ hóa cổng vào-ra (để CPU giao tiếp với hiết bi): - **Port-mapped I/O (Vào-ra riêng biệt)**

+ Mỗi thiết bị được gán một (hoặc một vài) **port** riêng. khối. Tốc độ ghi/xóa cao -> người dùng dễ dàng upgrade + Có thể cần **instruction riêng** cho I/O (ví dụ: `IN` OUT' trong x86). RAM: là bô nhớ khả biến, đơn vi lưu trữ là điên tích trong

Memory-mapped I/O (Vào-ra theo bản đồ bô nhớ) + Sử dụng **chung không gian địa chỉ bô nhớ. + Mỗi thiết bị được **ánh xạ (cấp)** một vùng địa ch

rong không gian địa chỉ bộ nhớ.

+ **Không cần thêm instruction riêng** cho I/O + Làm việc với thiết bị **giống như làm việc với bộ nhớ*

dùng các lệnh load/store thông thường, ví du: 'lw', 'sw' trong bởi tín hiệu đồng hồ bên ngoài.

* Giao tiến với thiết bị (được thực hiện thông qua Contr

egister và Data register trên I/O controller):

- **3 cơ chế giao tiếp**:

Polling (Programmed I/O - V-R bằng chương * CPU **chủ đông kiểm tra trang thái của thiết bi (

lụ: thiết bị đã sẵn sàng nhận/gửi dữ liệu chưa). * CPU sau đó **gửi yêu cầu** và **truyền/nhận d

* CPU phải **chò** trong suốt thời gian truy xuất thiê * Nhược điểm: Tổn thời gian CPH (khi chờ thiết hị châm

+ **Interrupt-driven (V-R bằng ngắt)**: * Thiết bị **chủ động thông báo trạng thái** cho CPU bằng cách **gửi tín hiệu ngắt**.

* CPU **tạm ngừ ng** chương trình hiện tại để xử lý ngắ chuyển đến **Interrupt Service Handler - ISH**).

* ISH xử lý yêu cầu của thiết bị (đọc/ghi dữ liêu) và sa đó CPU quay lại tiếp tục chương trình cũ.

* Ưu điểm: CPU không phải chờ thiết bị (khi thiết bị chi sẵn sàng).

Nhược điểm: Tốn chi phí xử lý ngắt.

* Dùng để truyền/nhận **khối lượng dữ liệu lớn** giữ hiết bi I/O và bộ nhớ chính **mà không cần sự tham gia trực iếp của CPU** trong quá trình truyền. * CPU chỉ cần **cấu hình** cho **DMA Controller

a thiết bị, địa chỉ đầu bộ nhớ, số lượng byte/word cần truyền). * DMA Controller tự thực hiện việc truyền dữ liệu. * Sau khi truyền xong, DMA Controller gửi **tín hiệ

ngắt** cho CPU để thông báo. * Ưu điểm: Giải phóng CPU trong quá trình truyền dữ liê

* Nhươc điểm: Có thể bị tấn công bằng mã độc (do qu rình đọc/ghi không được CPU quản lý). * Interrupt Service Handler (ISH): Doan code được thực thị khi CPU nhân tín hiệu ngắt để xi

ý yêu cầu của thiết bi. - Cách CPII đến được đoạn code xử lý interrunt:

+ Centralized dispatch

ho 0, lỗi truy cập bộ nhớ).

+ Vectored dispatch (CPU sử dụng số hiệu ngắt để tìm địa thỉ ISH trong bảng vector ngắt). * Interrupt (Ngắt):

 Khả năng tam ngừng chương trình hiện tại để thực th nương trình khác (thường là ISH).

 Xảy ra do các tín hiệu từ thiết bị phần cứng hoặc chươn rinh (exception). Các loại exception: interrupt, trap, fault, abort (ví dụ: chia

- Tìm hiểu các thiết bị nhập xuất hiện nay (bàn phím, chuộ nàn hình, ổ đĩa SSD/HDD, card mạng...).

- Tìm hiểu các loại bus thông dụng (PCI, PCIe, USB, SATA, ... HÊ THỐNG LƯU TRỮ

Về chức năng: thanh ghi, bô nhớ trong, bô nhớ ngoài. Theo vât lý: bô nhớ bán dẫn, bộ nhớ từ, bộ nhớ quang; có



Fhanh ghi: cấu tạo từ mạch lật, nằm trong CPU, có dụng lượng nhỏ nhất nhưng tốc độ truy xuất nhanh nhất, được dùng trong các bộ xử lý (lưu lệnh và dữ liệu). Có thể làm oằng nhiều công nghệ khác nhau: trigger, core, thin film. Thường tổ chức thành tập register file. Chứa các thông tin tam 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

EPROM: eraseable PROM, có thể viết nhưng phải xóa bằng tia cực tím sau khi ghi. EEPROM: electrically EPROM, xóa bằng điện, ghi/xóa từng

+ Sử dụng **không gian địa chỉ riêng** cho các thiết bị. | FlashROM: không thể xóa từng byte mà phải xóa từng | block do có nhiều nội dụng không cần thiết. Tùy vào cách | cần truy xuất khối lượng lớn dữ liệu, chỉ ngắt CPU một lần dung lirang

> tu điện, không phải hiệu điện thế nên giá thành thấp nhưng tốc đô truy xuất châm, tăng khoảng 7% mỗi năm, Gồm RAM tĩnh được chế tạo từ mạch lật và không cần làm tươi (SRAM) và RAM động được chế tạo từ tụ điện (DRAM). SDRAM: (synchronous RAM) việc truy xuất được đồng bộ

> Bô 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 hỏ

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 liệu -> có tốc độ gấp đội SDR (lý thuyết), chuyển được 64bit dữ liêu. DDRII gấp 2 lần DDRI. DDRIII gấp 34lần DDRI. Bô nhớ Cache: là mức thanh ghi gần nhất, dùng công nghệ

SRAM, có tốc đô cao hơn bộ nhớ chính. Đóng vai trò làn bộ nhớ đêm 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 vi từ nhớ, giữa cache và bố nhớ chính theo đơn vi block/line nhớ Nguyên lý hoat đông cache: cuc bô về thời gian (tempora

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 nằm dưới đầu đọc + **DMA (Direct Memory Access - Truy cập bộ nhớ trực thì những ô lân cận 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 nội dụng

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 penalty. 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 luong/loai cache. Tổ chức bô nhớ cache: phần tử gọi là line gồm data block hoặc điện từ trường. Phương pháp là sử dụng mã kiểm tra (4bytes) và tag chứa địa chỉ của block.

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

Fully Associate Mapping: một block có thể được nap vào bất kỳ line nào. Do 1 block có thể được nạp vào bất cứ line nào nên để xác định block đã nằm trong line hay chưa thì phải duyết tất cả các line -> chi phí tìm kiếm cao, dùng nhất hiện nay: USB, memory card, ROM. Không cần nguồn nhiều hịt cho địa chỉ tạo

Block/Tag s Vd: 24bit address, 222 blocks, block size = 4 bytes => w = 2bit & taa = 22hit

Direct Mapping: cải thiện nhược điểm của FAM bằng cách sử dụng hash table. Mỗi block Bị được ánh xa duy nhất vào môt line Li với Li = Bị mọd L với L là số line. Đơn giản nhanh, chi phí tổ chức thấp nhưng không uyển chuyểr (nếu truy xuất tuần tự các địa chỉ cùng 1 line thì liên tực cache miss dù cache còn trống nhiều chỗ; mỗi block cổ đinh 1 cache line).

r-s Vd: 24bit address, 222 blocks, 214 lines, block size = 4 bytes => w = 2, r = 14 & taa = 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 associate.

s-d he set d rd w Vd: 24bit address, 222 blocks, 214 lines, block size = 4bytes = w = 2, taa = 9, line = 13.

Chiến lược thay thể: không có lưa chon cho direct mapping. Với FAM và SAM: có thể chon random, FIFO, LRU (least recently used- dùng nhiều), LFU (least frequently used). Đồng bộ hóa cache và main mem: writethrough xảy ra

khi line/block bi 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 thế 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

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 bô xử lý) và off-chip cache (mức cao, nằm trên IC riêng và được nối với bô 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 lệnh và dữ liêu). Theo level có L1 (dùng riêng) và L2, L3 (dùng chung).

Thời gian truy xuất trung hình: AMAT = L1 hit-time + L1 miss-rate*L1 miss-penalty. Néu

có dùng L2 thì L1 miss-penalty = L2 hit-time + L2 missrate * 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

đô nhanh (vòng lặp), minimize lần cache miss trong mỗi Đĩa từ: thiết hị lưu trữ dữ liêu lâu dài nhổ hiến 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, phụ thuộc vào số track/mặt và tốc độ actuator. Rotation time: thời gian đĩa quay sao cho sector muốn đọc

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

Dung lượng 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 liêu lưu trên đĩa sẽ được lưu đồng thời trên tất cả các dĩa, Một số RAID thông dụng 0, 1 10, 5, 6, Cơ chế kiểm tra và sửa lỗi: lỗi xuất hiện do các xung điện

bằng cách thêm các bit 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 phổ biến do sư thay thế của flash memory. Flash memory: công nghệ lưu trữ được sử dụng phổ biến điện để duy trì dữ liệu. Có thể duy trì dữ liệu lâu dài, ít tốn điện năng. Nhưng số chu kỳ ghi/xóa bị giới hạn (do sự hao mòn), hầu hết khoảng trên 100 ngàn hoặc trên 1 triệu chu kỳ ghi/xóa dữ liêu.

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

Disk library: kết hợp đĩa cứng. Ontical jukehov: kết hơn đĩa quang

HÊ THỐNG NHẬP XUẤT

Vai trò: giao tiếp giữa thành phần xử lý của máy tính với các đối tương bên ngoài giúp các đối tượng này có thể cung cấp yêu cầu và dữ liệu cho thành phần xử lý.

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 loại bus liên kết hệ thống: điều khiển, dữ liêu, địa chỉ, 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 bô lênh của bô xử lý thường phải có một số lênh chuyên biệt để thao tác với thiết bị nhập xuất như IN và

hiết bị, sau đó gửi yêu cầu truy xuất và truyền data. CPU phải chờ trong suốt thời gian truy xuất thiết bi.

trình truy xuất thiết bị, thiết bị chủ động báo tình trạng với

cho một yêu cầu truy xuất dữ liêu.

| LDURB | 00111000010 | 11 | 450 | | D - format | | | |
|------------------------------|------------------------------|--------|--------|--------|-------------|--|--|--|
| B.cond | 01010100 | 8 | 672 | 679 | CB - format | | | |
| ORRI | 1011001000 | 10 | 712 | 713 | I - format | | | |
| EORI | 1101001000 | 10 | 840 | 841 | I - format | | | |
| STURH | 01111000000 | 11 | 960 | | D - format | | | |
| LDURH | 01111000010 | 11 | 962 | | D - format | | | |
| AND | 10001010000 | 11 | 1104 | | R - format | | | |
| ADD | 10001011000 | 11 | 1112 | | R - format | | | |
| ADDI | 1001000100 | 10 | 1160 | 1161 | I - format | | | |
| ANDI | 1001001000 | 10 | 1168 | 1169 | I - format | | | |
| BL | 100101 | 6 | 1184 | 1215 | B - format | | | |
| ORR | 10101010000 | 11 | 1360 | | R - format | | | |
| ADDS | 10101011000 | 11 | 1368 | | R - format | | | |
| ADDIS | 1011000100 | 10 | 1416 | 1417 | I - format | | | |
| CBZ | 10110100 | 8 | 1440 | 1447 | CB - format | | | |
| CBNZ | 10110101 | 8 | 1448 | 1455 | CB - format | | | |
| STURW | 10111000000 | 11 | 1472 | | D - format | | | |
| LDURSW | 10111000100 | 11 | 1476 | | D - format | | | |
| STXR | 11001000000 | 11 | 1600 | | D - format | | | |
| LDXR | 11001000010 | 11 | 1602 | | D - format | | | |
| EOR | 11101010000 | 11 | 1616 | | R - format | | | |
| SUB | 11001011000 | 11 | 1624 | | R - format | | | |
| SUBI | 1101000100 | 10 | 1672 | 1673 | I - format | | | |
| MOVZ | 110100101 | 9 | 1684 | 1687 | IM - format | | | |
| LSR | 11010011010 | 11 | 1690 | | R - format | | | |
| LSL | 11010011011 | 11 | 1691 | | R - format | | | |
| BR | 11010110000 | 11 | 1712 | | R - format | | | |
| ANDS | 11101010000 | 11 | 1872 | | R - format | | | |
| SUBS | 11101011000 | 11 | 1880 | | R - format | | | |
| SUBIS | 1111000100 | 10 | 1928 | 1929 | I - format | | | |
| ANDIS | 1111001000 | 10 | 1936 | 1937 | I - format | | | |
| MOVK | 111100101 | 9 | 1940 | 1943 | IM - format | | | |
| STUR | 11111000000 | 11 | 1984 | | D - format | | | |
| LDUR | 11111000010 | 11 | 1986 | | D - format | | | |
| CONG NGHE
EFF Shoat For T | Ví dụ cấu trúc lệnh R-Format | | | | | | | |
| | code | Rm | shamt | Rn | Rd | | | |
| | bits | 5 bits | 6 bits | 5 bits | 5 bits | | | |
| op | code | Rm | shamt | Rn | Rd | | | |





Cần càng nhiều bit cho toán hạng hằng số càng tốt immediate Rn 10 bits 12 bits 5 i opcode: mã thao tác, cho biết loại lệnh gì. Ro (Register source): chứa địa chỉ thanh ghi nguồn thứ 1 immediate: chức giá trị toán hang nguồn thứ 2. Bd (Register destination): chiva dia chi thanh chi dich

Cấu trúc lệnh hợp ngữ IM-Format Cấu trúc lệnh hợp ngữ MOVZ immediate opcode hw Tiêu chí phân loại: trên chức năng, đối tượng tương tác 110100101 01 0000000011111111 10011 A 0 1 F

645 1 Exit Else: SUB X19, X20, X21

4 0 0 0 0

Dung lượng cache = 256KB Kích thước 1 line = 1 block = 32 bytes

 Xác định cu thể số bịt cho 3 trường địa chỉ của X (W, L, T) nếu tổ chức theo kiểu direct mapping $RAM = 4GB = 2^32 bytes$

--> W dùng 5 bit Kích thước cache = 256KB = 2^18 byte

--> Số line trong cache = 2^18 / 2^5 = 2^13 lines --> L dùng 13 bit

+ memory-mapped; mỗi thiết bị được ánh xa vào một hoặc vài vùng nhớ. Truy xuất vào những vùng nhớ tương ứng với truy xuất vào thiết hị ví dụ như lw và sw của MIPS 3 cơ chế giao tiếp: + polling (còn goi là programmed I/O): CPU điều khiển toàn bộ quá trình nhập xuất, chủ động kiểm tra tình trang

+ interrupt-driven: CPU không phải chờ trong suốt quá

CPII tốn chi nhí xử lý ngắt direct memory access DMA: khi truyền dữ liêu lớn, phải

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

các nhóm thiết bị nhập xuất.

OUT của x86.

ngắt CPU nhiều lần (trên mỗi đơn vi truyền dữ liêu). Hê

--> Chiều dài địa chỉ N = 32 bit (W, L, T) Kích thước 1 line = 1 block = 32 bytes = 2^5 bytes

thống hỗ trợ DMA controller để giao tiếp với các thiết bị --> T = N - (W + L) = 32 - (5 + 13) = 14 bit