

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
Hanoi University of Science and Technology

KIẾN TRÚC MÁY TÍNH

Computer Architecture

Nguyễn Kim Khánh
Bộ môn Kỹ thuật máy tính
Viện Công nghệ thông tin và Truyền thông
Department of Computer Engineering (DCE)
School of Information and Communication Technology (SoICT)

Version: Jan 2014

Contact Information

- Address: 502-B1
- Mobile: 091-358-5533
- e-mail: khanhnk@soict.hust.edu.vn
khanh.nguyenkim@hust.edu.vn

Jan2014 Computer Architecture 2

Mục tiêu học phần

- Sinh viên được trang bị các kiến thức cơ sở về kiến trúc tập lệnh và tổ chức của máy tính, cũng như những vấn đề cơ bản trong thiết kế máy tính.
- Sau khi học xong học phần này, sinh viên có khả năng:
 - Tìm hiểu kiến trúc tập lệnh của các bộ xử lý cụ thể
 - Lập trình hợp ngữ trên một số kiến trúc
 - Đánh giá hiệu năng của các họ máy tính
 - Khai thác và quản trị hiệu quả các hệ thống máy tính
 - Phân tích và thiết kế máy tính

Jan2014 Computer Architecture 3

Tài liệu tham khảo chính

- [1] William Stallings - *Computer Organization and Architecture – Designing for Performance* – 2013 (9th edition)
- [2] David A. Patterson & John L. Hennessy - *Computer Organization and Design: The Hardware/Software Interface* – 2012 (revised 4th edition)
- [3] David Money Harris and Sarah L. Harris, *Digital Design and Computer Architecture* – 2013 (2nd edition)
- [4] Andrew S. Tanenbaum - *Structured Computer Organization* – 2012 (6th edition)

Jan2014 Computer Architecture 4

NKK-HUST

Nội dung học phần

- Chương 1. Giới thiệu chung
- Chương 2. Cơ bản về logic số
- Chương 3. Hệ thống máy tính
- Chương 4. Số học máy tính
- Chương 5. Kiến trúc tập lệnh
- Chương 6. Bộ xử lý
- Chương 7. Bộ nhớ máy tính
- Chương 8. Hệ thống vào-ra
- Chương 9. Các kiến trúc song song

Jan2014 Computer Architecture 5

NKK-HUST

Chú ý: Bài giảng mới nhất Jan 2014

<ftp://dce.hust.edu.vn/khanhnk/CA>

Jan2014 Computer Architecture 6

NKK-HUST

Kiến trúc máy tính

Chương 1 GIỚI THIỆU CHUNG

Nguyễn Kim Khánh
Trường Đại học Bách khoa Hà Nội

Jan2014 Computer Architecture 7

NKK-HUST

Nội dung

- 1.1. Máy tính và phân loại
- 1.2. Khái niệm kiến trúc máy tính
- 1.3. Sự tiến hóa của máy tính
- 1.4. Hiệu năng máy tính

Jan2014 Computer Architecture 8

1.1. Máy tính và phân loại máy tính

1. Máy tính

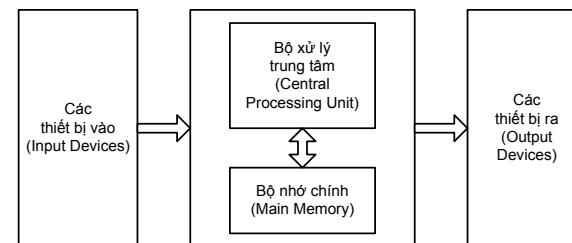
- **Máy tính (Computer)** là thiết bị điện tử thực hiện các công việc sau:
 - Nhận thông tin vào,
 - Xử lý thông tin theo dãy các lệnh được nhớ sẵn bên trong,
 - Đưa thông tin ra.
 - Dãy các lệnh nằm trong bộ nhớ để yêu cầu máy tính thực hiện công việc cụ thể gọi là **chương trình (program)**
- Máy tính hoạt động theo chương trình.

Jan2014

Computer Architecture

9

Máy tính



Jan2014

Computer Architecture

10

2. Phân loại máy tính

- Phân loại truyền thống:
 - Máy vi tính (Microcomputers)
 - Máy tính nhỏ (Minicomputers)
 - Máy tính lớn (Mainframe Computers)
 - Siêu máy tính (Supercomputers)

Jan2014

Computer Architecture

11

Phân loại máy tính hiện đại [P&H]

- Thiết bị di động cá nhân (Personal Mobile Devices):
 - Smartphones, Tablet
- Máy tính cá nhân đa dụng (Personal Computers)
 - Desktop computers, Laptop computers
- Máy chủ (Servers)
 - Thực chất là Máy phục vụ
 - Dùng trong mạng theo mô hình Client/Server
- Máy tính cụm/máy tính qui mô lớn (Clusters/Warehouse Scale Computers):
 - Sử dụng tại các trung tâm tính toán, trung tâm dữ liệu
 - Supercomputers
- Máy tính nhúng (Embedded Computers)
 - Đặt ẩn trong thiết bị khác
 - Được thiết kế chuyên dụng

Jan2014

Computer Architecture

12

1.2. Khái niệm kiến trúc máy tính

- Định nghĩa trước đây về kiến trúc máy tính:
 - Là thiết kế **kiến trúc tập lệnh** (Instruction Set Architecture – ISA)
 - Các thuộc tính của máy tính theo cách nhìn người lập trình (hardware/software interface)
 - Là định nghĩa hẹp

Jan2014

Computer Architecture

13

Định nghĩa của Hennessy/ Patterson

- Kiến trúc máy tính bao gồm:
 - **Kiến trúc tập lệnh** (Instruction Set Architecture): nghiên cứu máy tính theo cách nhìn của người lập trình (hardware/software interface).
 - **Tổ chức máy tính** (Computer Organization) hay **Vi kiến trúc** (Microarchitecture): nghiên cứu thiết kế máy tính ở mức cao, chẳng hạn như hệ thống nhớ, cấu trúc bus, thiết kế bên trong CPU.
 - **Phần cứng** (Hardware): nghiên cứu thiết kế logic chi tiết và công nghệ đóng gói của máy tính.
- **Kiến trúc tập lệnh thay đổi chậm, tổ chức và phần cứng máy tính thay đổi rất nhanh.**

Jan2014

Computer Architecture

14

Kiến trúc tập lệnh

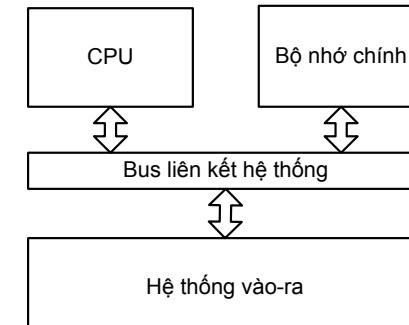
- Kiến trúc tập lệnh của máy tính bao gồm:
- **Tập lệnh**: tập hợp các chuỗi số nhị phân mã hoá cho các thao tác mà máy tính có thể thực hiện
 - **Các kiểu dữ liệu**: các kiểu dữ liệu mà máy tính có thể xử lý

Jan2014

Computer Architecture

15

Cấu trúc cơ bản của máy tính



Jan2014

Computer Architecture

16

Các thành phần cơ bản của máy tính

- **Bộ xử lý trung tâm** (Central Processing Unit): Điều khiển hoạt động của máy tính và xử lý dữ liệu.
- **Bộ nhớ chính** (Main Memory): Chứa các chương trình và dữ liệu đang được sử dụng.
- **Hệ thống vào-ra** (Input/Output System): Trao đổi thông tin giữa máy tính với bên ngoài.
- **Bus liên kết hệ thống** (System Interconnection Bus): Kết nối và vận chuyển thông tin giữa các thành phần với nhau.

Jan2014

Computer Architecture

17

Mô hình phân lớp của máy tính

Phần mềm ứng dụng
Phần mềm trung gian
Hệ điều hành
Kiến trúc tập lệnh
Vi kiến trúc
Logic-số
Mạch điện tử

- **Phần cứng (Hardware)**: hệ thống vật lý của máy tính.
- **Phần mềm (Software)**: các chương trình và dữ liệu.

Jan2014

Computer Architecture

18

1.3. Sự tiến hóa của máy tính

Các thế hệ máy tính

- Thế hệ thứ nhất: Máy tính dùng **đèn điện tử** chân không (1950s)
- Thế hệ thứ hai: Máy tính dùng **transistor** (1960s)
- Thế hệ thứ ba: Máy tính dùng **vi mạch SSI, MSI và LSI** (1970s)
- Thế hệ thứ tư: Máy tính dùng **vi mạch VLSI** (1980s)
- Thế hệ thứ năm: Máy tính dùng **vi mạch ULSI, SoC** (1990s-nay)

Jan2014

Computer Architecture

19

ENIAC – Máy tính điện tử đầu tiên

- Electronic Numerical Integator and Computer
- Dự án của Bộ Quốc phòng Mỹ
- Do John Mauchly và John Presper Eckert ở Đại học Pennsylvania thiết kế.
- Bắt đầu từ 1943, hoàn thành 1946
- Nặng 30 tấn
- 18000 đèn điện tử và 1500 rơle
- 5000 phép cộng/giây
- Xử lý theo số thập phân
- Bộ nhớ chỉ lưu trữ dữ liệu
- Lập trình bằng cách thiết lập vị trí của các chuyển mạch và các cáp nối.

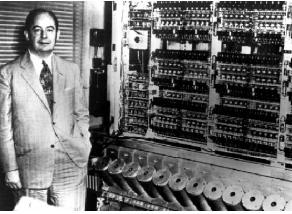
Jan2014

Computer Architecture

20

Máy tính von Neumann

- Chính là máy tính IAS (thực hiện tại Princeton Institute for Advanced Studies)
- Bắt đầu 1947, hoàn thành 1952
- Do John von Neumann thiết kế
- Được xây dựng theo ý tưởng “chương trình được lưu trữ” - (*stored-program concept*) của von Neumann/Turing (1945)
- Trở thành mô hình cơ bản của máy tính



Jan2014 Computer Architecture 21

Vi mạch

- Vi mạch hay là mạch tích hợp (Integrated Circuit - IC): mạch điện tử gồm nhiều transistors và các linh kiện khác được tích hợp trên một chip bán dẫn.
- Phân loại vi mạch theo qui mô tích hợp:
 - SSI - Small Scale Integration
 - MSI - Medium Scale Integration
 - LSI - Large Scale Integration
 - VLSI - Very Large Scale Integration
 - ULSI - Ultra Large Scale Integration

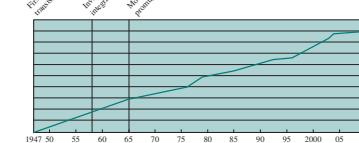
Jan2014 Computer Architecture 22

Một số vi mạch số điển hình

- **Bộ vi xử lý (Microprocessors):** CPU được chế tạo trên một chip.
- **Vi mạch điều khiển tổng hợp (Chipset):** một hoặc một vài vi mạch thực hiện được nhiều chức năng điều khiển và nối ghép.
- **Bộ nhớ bán dẫn (Semiconductor Memory):** ROM, RAM, Flash
- **Hệ thống trên chip (SoC – System on Chip)**
- **Các bộ vi điều khiển (Microcontrollers)**

Jan2014 Computer Architecture 23

Luật Moore



Year	Transistors (approx.)
1950	1
1960	10
1970	100
1980	1000
1990	10,000
2000	100,000
2010	1,000,000
2020	10,000,000

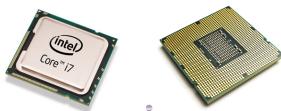
- Gordon Moore – người đồng sáng lập Intel
- Số transistors trên chip sẽ gấp đôi sau 18 tháng
- Giá thành của chip hầu như không thay đổi
- Mật độ cao hơn, do vậy đường dẫn ngắn hơn
- Kích thước nhỏ hơn dẫn tới độ phức tạp tăng lên
- Điện năng tiêu thụ ít hơn
- Hệ thống có ít các chip liên kết với nhau, do đó tăng độ tin cậy

Jan2014 Computer Architecture 24

NKK-HUST

Sự phát triển của vi xử lý

- 1971: bộ vi xử lý 4-bit Intel 4004
- 1972: các bộ xử lý 8-bit
- 1978: các bộ xử lý 16-bit
- 1985: các bộ xử lý 32-bit
- 2001: các bộ xử lý 64-bit
- 2006: các bộ xử lý đa lõi (multicores)



Jan2014 Computer Architecture 25

NKK-HUST

Máy tính ngày nay



Jan2014 Computer Architecture 26

NKK-HUST

1.4. Hiệu năng máy tính

- Định nghĩa hiệu năng P(Performance):

$$P = 1/t$$

trong đó: t là thời gian thực hiện
- “Máy tính A nhanh hơn máy B n lần”

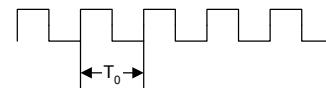
$$P_A / P_B = t_B / t_A = n$$
- Ví dụ: Thời gian chạy chương trình:
 - 10s trên máy A, 15s trên máy B
 - $t_B / t_A = 15s / 10s = 1.5$
 - Vậy máy A nhanh hơn máy B 1.5 lần

Jan2014 Computer Architecture 27

NKK-HUST

Xung nhịp của CPU

- Hoạt động của CPU được điều khiển bởi xung nhịp có tần số xác định



- Chu kỳ xung nhịp T_0 (Clock period): thời gian của một chu kỳ
- Tần số xung nhịp f_0 (Clock rate): số chu kỳ trong 1 giây.

$$f_0 = 1/T_0$$
- VD: Bộ xử lý có $f_0 = 4\text{GHz} = 4000\text{MHz} = 4 \times 10^9\text{Hz}$

$$T_0 = 1/(4 \times 10^9) = 0.25 \times 10^{-9}\text{s} = 0.25\text{ns}$$

Jan2014 Computer Architecture 28

Thời gian CPU (t_{CPU})

$$t_{CPU} = n \times T_0 = \frac{n}{f_0}$$

- trong đó: n là số chu kỳ xung nhịp
- Hiệu năng được tăng lên bằng cách:
 - Giảm số chu kỳ xung nhịp n
 - Tăng tần số xung nhịp f_0

Jan2014

Computer Architecture

29

Ví dụ

- Máy tính A:
 - Tần số xung nhịp: $f_A = 2\text{GHz}$
 - Thời gian của CPU: $t_A = 10\text{s}$
- Máy tính B
 - Thời gian của CPU: $t_B = 6\text{s}$
 - Số chu kỳ xung nhịp của B = $1.2 \times$ Số chu kỳ xung nhịp của A
- Xác định tần số xung nhịp của máy B (f_B)?

Jan2014

Computer Architecture

30

Ví dụ

- Máy tính A:
 - Tần số xung nhịp: $f_A = 2\text{GHz}$
 - Thời gian của CPU: $t_A = 10\text{s}$
- Máy tính B
 - Thời gian của CPU: $t_B = 6\text{s}$
 - Số chu kỳ xung nhịp của B = $1.2 \times$ Số chu kỳ xung nhịp của A
- Xác định tần số xung nhịp của máy B (f_B)?
- Giải:

$$f_B = \frac{n_B}{t_B} = \frac{1.2 \times n_A}{6s}$$

$$n_A = t_A \times f_A = 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

$$f_B = \frac{1.2 \times 20 \times 10^9}{6s} = \frac{24 \times 10^9}{6s} = 4\text{GHz}$$

Jan2014

Computer Architecture

31

Số lệnh và số chu kỳ trên một lệnh

- Số chu kỳ = Số lệnh x Số chu kỳ trên một lệnh

$$n = IC \times CPI$$

n - số chu kỳ, IC - số lệnh (Instruction Count), CPI - số chu kỳ trên một lệnh (Cycles per Instruction)

- Thời gian thực hiện của CPU:

$$t_{CPU} = IC \times CPI \times T_0 = \frac{IC \times CPI}{f_0}$$

- Trong trường hợp các lệnh khác nhau có CPI khác nhau, cần tính CPI trung bình

Jan2014

Computer Architecture

32

Ví dụ

- Máy tính A: $T_A = 250\text{ps}$, $CPI_A = 2.0$
- Máy tính B: $T_B = 500\text{ps}$, $CPI_B = 1.2$
- Cùng kiến trúc tập lệnh (ISA)
- Máy nào nhanh hơn và nhanh hơn bao nhiêu ?

Jan2014

Computer Architecture

33

Ví dụ

- Máy tính A: $T_A = 250\text{ps}$, $CPI_A = 2.0$
- Máy tính B: $T_B = 500\text{ps}$, $CPI_B = 1.2$
- Cùng kiến trúc tập lệnh (ISA)
- Máy nào nhanh hơn và nhanh hơn bao nhiêu ?

$$\begin{aligned} t_A &= IC \times CPI_A \times T_A \\ &= IC \times 2.0 \times 250\text{ps} = IC \times 500\text{ps} \end{aligned}$$

$$\begin{aligned} t_B &= IC \times CPI_B \times T_B \\ &= IC \times 1.2 \times 500\text{ps} = IC \times 600\text{ps} \end{aligned}$$

$$\frac{t_B}{t_A} = \frac{IC \times 600\text{ps}}{IC \times 500\text{ps}} = 1.2$$

Vậy:
A nhanh hơn B 1.2 lần

Jan2014

Computer Architecture

34

Chi tiết hơn về CPI

- Nếu loại lệnh khác nhau có số chu kỳ khác nhau, ta có tổng số chu kỳ:

$$n = \sum_{i=1}^K (CPI_i \times IC_i)$$

- CPI trung bình:

$$CPI_{TB} = \frac{n}{IC} = \sum_{i=1}^K \left(CPI_i \times \frac{IC_i}{IC} \right)$$

Jan2014

Computer Architecture

35

Ví dụ

- Cho bảng chỉ ra các dãy lệnh sử dụng các lệnh thuộc các loại A, B, C. Tính CPI trung bình?

Loại lệnh	A	B	C
CPI theo loại lệnh	1	2	3
IC trong dãy lệnh 1	2	1	2
IC trong dãy lệnh 2	4	1	1

Jan2014

Computer Architecture

36

Ví dụ

- Cho bảng chỉ ra các dãy lệnh sử dụng các lệnh thuộc các loại A, B, C. Tính CPI trung bình?

Loại lệnh	A	B	C
CPI theo loại lệnh	1	2	3
IC trong dãy lệnh 1	2	1	2
IC trong dãy lệnh 2	4	1	1

- Dãy lệnh 1: IC = 5
 - Số chu kỳ
= $2 \times 1 + 1 \times 2 + 2 \times 3$
= 10
 - $CPI_{TB} = 10/5 = 2.0$
- Dãy lệnh 2: IC = 6
 - Số chu kỳ
= $4 \times 1 + 1 \times 2 + 1 \times 3$
= 9
 - $CPI_{TB} = 9/6 = 1.5$

Jan2014

Computer Architecture

37

Tóm tắt về Hiệu năng

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

$$t_{CPU} = IC \times CPI \times T_0 = \frac{IC \times CPI}{f_0}$$

- Hiệu năng phụ thuộc vào:
 - Thuật toán
 - Ngôn ngữ lập trình
 - Chương trình dịch
 - Kiến trúc tập lệnh

Jan2014

Computer Architecture

38

MIPS như là thước đo hiệu năng

- MIPS: Millions of Instructions Per Second (Số triệu lệnh trên 1 giây)

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} = \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI} \times 10^6}{\text{Clock rate}}} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

$$\text{MIPS} = \frac{f_0}{\text{CPI} \times 10^6} \quad \text{CPI} = \frac{f_0}{\text{MIPS} \times 10^6}$$

Jan2014

Computer Architecture

39

Ví dụ

Tính MIPS của bộ xử lý với:
clock rate = 2GHz và CPI = 4

Jan2014

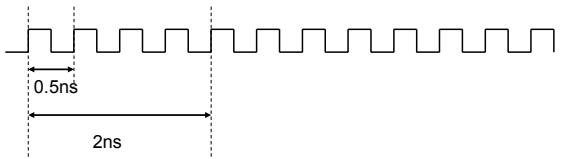
Computer Architecture

40

NKK-HUST

Ví dụ

Tính MIPS của bộ xử lý với:
clock rate = 2GHz và CPI = 4



1 chu kỳ = $1/(2 \times 10^9) = 0,5\text{ns}$
 CPI = 4 → thời gian thực hiện 1 lệnh:
 $4 \times 0,5\text{ns} = 2\text{ns}$
 Vậy bộ xử lý thực hiện được 500 MIPS

Jan2014 Computer Architecture 41

NKK-HUST

Ví dụ

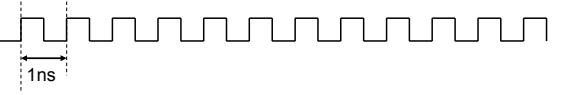
Tính CPI của bộ xử lý với:
clock rate = 1GHz và 400 MIPS?

Jan2014 Computer Architecture 42

NKK-HUST

Ví dụ

Tính CPI của bộ xử lý với:
clock rate = 1GHz và 400 MIPS?



4x10⁸ lệnh thực hiện trong 1s
 → 1 lệnh thực hiện trong $1/(4 \times 10^8)\text{s} = 2,5\text{ns}$
 → CPI = 2,5

Jan2014 Computer Architecture 43

NKK-HUST

MFLOPS

Millions of Floating Point Operations per Second
 (Số triệu phép toán số dấu phẩy động trên một giây)

$$\text{MFLOPS} = \frac{\text{Executed floating point operations}}{\text{Execution time} \times 10^6}$$

GFLOPS (10^9)
 TFLOPS (10^{12})

Jan2014 Computer Architecture 44



Hết chương 1

Jan2014 Computer Architecture 45



Kiến trúc máy tính

Chương 2 CƠ BẢN VỀ LOGIC SỐ

Nguyễn Kim Khánh
Trường Đại học Bách khoa Hà Nội

Jan2014 Computer Architecture 46



Nội dung học phần

- Chương 1. Giới thiệu chung
- Chương 2. Cơ bản về logic số
- Chương 3. Hệ thống máy tính
- Chương 4. Số học máy tính
- Chương 5. Kiến trúc tập lệnh
- Chương 6. Bộ xử lý
- Chương 7. Bộ nhớ máy tính
- Chương 8. Hệ thống vào-ra
- Chương 9. Các kiến trúc song song

Jan2014 Computer Architecture 47



Nội dung của chương 2

- 2.1. Các hệ đếm cơ bản
- 2.2. Đại số Boole
- 2.3. Các cổng logic
- 2.4. Mạch tổ hợp
- 2.5. Mạch dãy

Jan2014 Computer Architecture 48

NKK-HUST

2.1. Các hệ đếm cơ bản

- Hệ thập phân (Decimal System)
→ **con người sử dụng**
- Hệ nhị phân (Binary System)
→ **máy tính sử dụng**
- Hệ mươi sáu (Hexadecimal System)
→ dùng để viết gọn cho số nhị phân

Jan2014 Computer Architecture 49

NKK-HUST

1. Hệ thập phân

- Cơ số 10
- 10 chữ số: 0,1,2,3,4,5,6,7,8,9
- Dùng n chữ số thập phân có thể biểu diễn được 10^n giá trị khác nhau:
 - 00...000 = 0
 - 99...999 = $10^n - 1$

Jan2014 Computer Architecture 50

NKK-HUST

Dạng tổng quát của số thập phân

$$A = a_n a_{n-1} \dots a_1 a_0, a_{-1} \dots a_{-m}$$

Giá trị của A được hiểu như sau:

$$A = a_n 10^n + a_{n-1} 10^{n-1} + \dots + a_1 10^1 + a_0 10^0 + a_{-1} 10^{-1} + \dots + a_{-m} 10^{-m}$$

$$A = \sum_{i=-m}^n a_i 10^i$$

Jan2014 Computer Architecture 51

NKK-HUST

Ví dụ số thập phân

$$472.38 = 4 \times 10^2 + 7 \times 10^1 + 2 \times 10^0 + 3 \times 10^{-1} + 8 \times 10^{-2}$$

- Các chữ số của phần nguyên:
 - $472 : 10 = 47$ dư 2 ↑
 - $47 : 10 = 4$ dư 7 ↓
 - $4 : 10 = 0$ dư 4 ↓
- Các chữ số của phần lẻ:
 - $0.38 \times 10 = 3.8$ phần nguyên = 3 ↓
 - $0.8 \times 10 = 8.0$ phần nguyên = 8 ↓

Jan2014 Computer Architecture 52

NKK-HUST

2. Hệ nhị phân

- Cơ số 2
- 2 chữ số nhị phân: 0 và 1
- chữ số nhị phân gọi là **bit** (binary digit)
- Bit là đơn vị thông tin nhỏ nhất
- Dùng n bit có thể biểu diễn được 2^n giá trị khác nhau:
 - 00...000 = 0
 - 11...111 = $2^n - 1$

Jan2014 Computer Architecture 53

NKK-HUST

Bits, Bytes, Nibbles...

- Bits

10010110
most significant bit least significant bit
byte

- Bytes & Nibbles

10010110
byte nibble

- Bytes

CEBF9AD7
most significant byte least significant byte

Jan2014 Computer Architecture 54

NKK-HUST

Lũy thừa hai

- $2^{10} = 1$ kilo ≈ 1000 (1024)
- $2^{20} = 1$ mega ≈ 1 triệu (1,048,576)
- $2^{30} = 1$ giga ≈ 1 tỷ (1,073,741,824)
- $2^{40} = 1$ tera ≈ 1000 tỷ
- $2^{50} = 1$ peta ≈ 1 triệu tỷ

Jan2014 Computer Architecture 55

NKK-HUST

Dạng tổng quát của số nhị phân

Có một số nhị phân A như sau:

$$A = a_n a_{n-1} \dots a_1 a_0, a_{-1} \dots a_{-m}$$

Giá trị của A được tính như sau:

$$A = a_n 2^n + a_{n-1} 2^{n-1} + \dots + a_1 2^1 + a_0 2^0 + a_{-1} 2^{-1} + \dots + a_{-m} 2^{-m}$$

$$A = \sum_{i=-m}^n a_i 2^i$$

Jan2014 Computer Architecture 56

NKK-HUST

Ví dụ số nhị phân

$$\begin{aligned}
 1101001.1011_{(2)} &= \\
 &= 2^6 + 2^5 + 2^3 + 2^0 + 2^{-1} + 2^{-3} + 2^{-4} \\
 &= 64 + 32 + 8 + 1 + 0.5 + 0.125 + 0.0625 \\
 &= 105.6875_{(10)}
 \end{aligned}$$

Jan2014 Computer Architecture 57

NKK-HUST

Chuyển đổi số nguyên thập phân sang nhị phân

- Phương pháp 1: chia dần cho 2 rồi lấy phần dư
- Phương pháp 2: Phân tích thành tổng của các số 2^i → nhanh hơn

Jan2014 Computer Architecture 58

NKK-HUST

Phương pháp chia dần cho 2

- Ví dụ: chuyển đổi $105_{(10)}$
 - $105 : 2 = 52$ dư 1
 - $52 : 2 = 26$ dư 0
 - $26 : 2 = 13$ dư 0
 - $13 : 2 = 6$ dư 1
 - $6 : 2 = 3$ dư 0
 - $3 : 2 = 1$ dư 1
 - $1 : 2 = 0$ dư 1
- Kết quả: $105_{(10)} = 1101001_{(2)}$

Jan2014 Computer Architecture 59

NKK-HUST

Phương pháp phân tích thành tổng của các 2^i

- Ví dụ 1: chuyển đổi $105_{(10)}$
 - $105 = 64 + 32 + 8 + 1 = 2^6 + 2^5 + 2^3 + 2^0$
- Để biểu diễn số $105_{(10)}$ bằng nhị phân, ta cần tìm các số 2^i sao cho tổng của chúng bằng $105_{(10)}$. Các số 2^i cần tìm là $2^6, 2^5, 2^3, 2^0$.
- Kết quả: $105_{(10)} = 0110 1001_{(2)}$
- Ví dụ 2: $17000_{(10)} = 16384 + 512 + 64 + 32 + 8 = 2^{14} + 2^9 + 2^6 + 2^5 + 2^3$
- Biểu diễn số $17000_{(10)}$ bằng nhị phân:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	1	1	0	1	0	0	1

$$\begin{aligned}
 17000_{(10)} &= 0100\ 0010\ 0110\ 1000_{(2)} \\
 &\quad \text{15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0}
 \end{aligned}$$

Jan2014 Computer Architecture 60

NKK-HUST

Chuyển đổi số lẻ thập phân sang nhị phân

- Ví dụ 1: chuyển đổi $0.6875_{(10)}$
 - $0.6875 \times 2 = 1.375$ phần nguyên = 1
 - $0.375 \times 2 = 0.75$ phần nguyên = 0
 - $0.75 \times 2 = 1.5$ phần nguyên = 1
 - $0.5 \times 2 = 1.0$ phần nguyên = 1
- Kết quả : $0.6875_{(10)} = 0.1011_{(2)}$

Jan2014

Computer Architecture

61

NKK-HUST

Chuyển đổi số lẻ thập phân sang nhị phân (tiếp)

- Ví dụ 2: chuyển đổi $0.81_{(10)}$
 - $0.81 \times 2 = 1.62$ phần nguyên = 1
 - $0.62 \times 2 = 1.24$ phần nguyên = 1
 - $0.24 \times 2 = 0.48$ phần nguyên = 0
 - $0.48 \times 2 = 0.96$ phần nguyên = 0
 - $0.96 \times 2 = 1.92$ phần nguyên = 1
 - $0.92 \times 2 = 1.84$ phần nguyên = 1
 - $0.84 \times 2 = 1.68$ phần nguyên = 1
- $0.81_{(10)} \approx 0.1100111_{(2)}$

Jan2014

Computer Architecture

62

NKK-HUST

Chuyển đổi số lẻ thập phân sang nhị phân (tiếp)

- Ví dụ 3: chuyển đổi $0.2_{(10)}$
 - $0.2 \times 2 = 0.4$ phần nguyên = 0
 - $0.4 \times 2 = 0.8$ phần nguyên = 0
 - $0.8 \times 2 = 1.6$ phần nguyên = 1
 - $0.6 \times 2 = 1.2$ phần nguyên = 1
 - $0.2 \times 2 = 0.4$ phần nguyên = 0
 - $0.4 \times 2 = 0.8$ phần nguyên = 0
 - $0.8 \times 2 = 1.6$ phần nguyên = 1
 - $0.6 \times 2 = 1.2$ phần nguyên = 1
- $0.2_{(10)} \approx 0.00110011_{(2)}$

Jan2014

Computer Architecture

63

NKK-HUST

3. Hệ mười sáu (Hexa)

- Cơ số 16
- 16 chữ số: 0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F
- Dùng để viết gọn cho số nhị phân: cứ một nhóm 4-bit sẽ được thay bằng một chữ số Hexa

Jan2014

Computer Architecture

64

NKK-HUST

Quan hệ giữa số nhị phân và số Hexa

4-bit	Chữ số Hexa
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Ví dụ chuyển đổi số nhị phân → số Hexa:

- $1011\ 0011_2 = B3_{16}$
- $0000\ 0000_2 = 00_{16}$
- $0010\ 1101\ 1001\ 1010_2 = 2D9A_{16}$
- $1111\ 1111\ 1111\ 1111_2 = FFFF_{16}$

Computer Architecture 65

NKK-HUST

2.2. Đại số Boole

- Đại số Boole sử dụng các biến logic và phép toán logic
- Biến logic có thể nhận giá trị 1 (TRUE) hoặc 0 (FALSE)
- Phép toán logic cơ bản là AND, OR và NOT với ký hiệu như sau:
 - A AND B : $A \cdot B$
 - A OR B : $A + B$
 - NOT A : \bar{A}
- Thứ tự ưu tiên: NOT > AND > OR

Jan2014 Computer Architecture 66

NKK-HUST

Các phép toán logic (tiếp)

- Các phép toán NAND, NOR, XOR:
 - A NAND B: $\overline{A \cdot B}$
 - A NOR B : $\overline{A + B}$
 - A XOR B: $A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B$

Computer Architecture 67

NKK-HUST

Phép toán đại số Boole

A	B	NOT A \bar{A}	A AND B $A \cdot B$	A OR B $A + B$	A NAND B $\overline{A \cdot B}$	A NOR B $\overline{A + B}$	A XOR B $A \oplus B$
0	0	1	0	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	1	0	1
1	1	0	1	1	0	0	0

Computer Architecture 68

Các đồng nhất thức của đại số Boole	
$A \cdot B = B \cdot A$	$A + B = B + A$
$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$	$A + (B \cdot C) = (A + B) \cdot (A + C)$
$1 \cdot A = A$	$0 + A = A$
$A \cdot \bar{A} = 0$	$A + \bar{A} = 1$
$0 \cdot A = 0$	$1 + A = 1$
$A \cdot A = A$	$A + A = A$
$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$
$\overline{A \cdot B} = \bar{A} + \bar{B}$ (Định lý De Morgan)	$\overline{A + B} = \bar{A} \cdot \bar{B}$ (Định lý De Morgan)

Jan2014

Computer Architecture

69

2.3. Các cổng logic (Logic Gates)	
■ Thực hiện các hàm logic:	<ul style="list-style-type: none"> ▪ NOT, AND, OR, NAND, NOR, etc.
■ Cổng logic một đầu vào:	<ul style="list-style-type: none"> ▪ Cổng NOT, bộ đệm (buffer)
■ Cổng hai đầu vào:	<ul style="list-style-type: none"> ▪ AND, OR, XOR, NAND, NOR, XNOR
■ Cổng nhiều đầu vào	

Jan2014

Computer Architecture

70

Các cổng logic																		
Name	Graphical Symbol	Algebraic Function	Truth Table															
AND		$F = A \cdot B$ or $F = AB$	<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>F</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1
A	B	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = A + B$	<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>F</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	1
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT		$F = \bar{A}$ or $F = A'$	<table border="1"> <thead> <tr> <th>A</th><th>F</th></tr> </thead> <tbody> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td></tr> </tbody> </table>	A	F	0	1	1	0									
A	F																	
0	1																	
1	0																	
NAND		$F = \overline{AB}$	<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>F</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	F	0	0	1	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = \overline{A + B}$	<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>F</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	0
A	B	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
XOR		$F = A \oplus B$	<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>F</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																

Jan2014

Computer Architecture

71

Tập đầy đủ	
■ Là tập các cổng có thể thực hiện được bất kỳ hàm logic nào từ các cổng của tập đó.	
■ Một số ví dụ về tập đầy đủ:	
<ul style="list-style-type: none"> ▪ {AND, OR, NOT} ▪ {AND, NOT} ▪ {OR, NOT} ▪ {NAND} ▪ {NOR} 	

Jan2014

Computer Architecture

72

NKK-HUST

Sử dụng cỗng NAND

The block shows four examples of logic function implementation:

- A single NAND gate with inputs A and B, output \bar{A} .
- A two-input NAND gate followed by an inverter to implement $A \oplus B$. The first NAND gate has inputs A and B, outputting $\bar{A} \cdot \bar{B}$. This is then fed into a second inverter to produce the final output $A \oplus B$.
- A two-input NAND gate followed by an inverter to implement \bar{A} . The first NAND gate has inputs A and B, outputting $\bar{A} \cdot \bar{B}$. This is then fed into a second inverter to produce the final output \bar{A} .
- A two-input NAND gate followed by an inverter to implement $A + B$. The first NAND gate has inputs A and B, outputting $\bar{A} \cdot \bar{B}$. This is then fed into a second inverter to produce the final output $A + B$.

Jan2014 Computer Architecture 73

NKK-HUST

Sử dụng cỗng NOR

The block shows four examples of logic function implementation:

- A single NOR gate with inputs A and B, output \bar{A} .
- A two-input NOR gate followed by an inverter to implement $A + B$. The first NOR gate has inputs A and B, outputting $(\bar{A} + \bar{B})$. This is then fed into a second inverter to produce the final output $A + B$.
- A two-input NOR gate followed by an inverter to implement \bar{A} . The first NOR gate has inputs A and B, outputting $(\bar{A} + \bar{B})$. This is then fed into a second inverter to produce the final output \bar{A} .
- A two-input NOR gate followed by an inverter to implement $A \cdot B$. The first NOR gate has inputs A and B, outputting $(\bar{A} + \bar{B})$. This is then fed into a second inverter to produce the final output $A \cdot B$.

Jan2014 Computer Architecture 74

NKK-HUST

Một số ví dụ vi mạch logic

The block displays six integrated circuit packages, each with its pinout diagram and part number:

- Top row: 7400 (quad 2-input AND gate)
- Middle row: 7402 (quad 2-input OR gate)
- Bottom row: 7411 (quad 2-input NOR gate), 7422 (quad 2-input XOR gate), 7432 (quad 2-input XNOR gate), and 7480 (hex inverter)

Jan2014 Computer Architecture 75

NKK-HUST

2.4. Mạch tổ hợp

- Mạch logic là mạch bao gồm:**
 - Các đầu vào (Inputs)
 - Các đầu ra (Outputs)
 - Đặc tả chức năng (Functional specification)
 - Đặc tả thời gian (Timing specification)
- Các kiểu mạch logic:**
 - Mạch logic tổ hợp (Combinational Logic)
 - Mạch không nhớ
 - Đầu ra được xác định bởi các giá trị hiện tại của đầu vào
 - Mạch logic dây (Sequential Logic)
 - Mạch có nhớ
 - Đầu ra được xác định bởi các giá trị trước đó và giá trị hiện tại của đầu vào

Jan2014 Computer Architecture 76

Mạch tổ hợp

- Mạch tổ hợp là mạch logic trong đó đầu ra chỉ phụ thuộc đầu vào ở thời điểm hiện tại.
- Là mạch không nhớ và được thực hiện bằng các cổng logic cơ bản
- Mạch tổ hợp có thể được định nghĩa theo ba cách:
 - Bảng thật
 - Dạng sơ đồ
 - Phương trình Boole

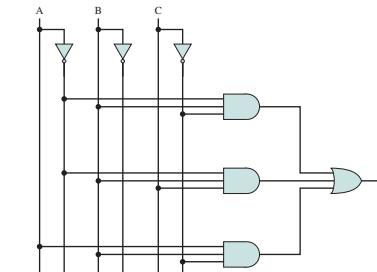
Jan2014

Computer Architecture

77

Ví dụ

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0



$$F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}$$

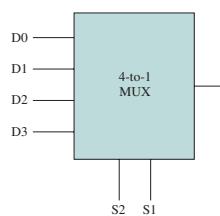
Jan2014

Computer Architecture

78

Bộ dồn kênh (Multiplexer-MUX)

- 2^n đầu vào dữ liệu
- n đầu vào chọn
- 1 đầu ra
- Đầu vào chọn (S) xác định đầu vào nào (D) sẽ được nối với đầu ra (F).



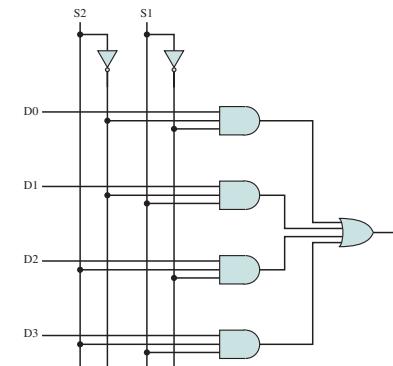
S2	S1	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Jan2014

Computer Architecture

79

Thực hiện MUX bốn đầu vào



Jan2014

Computer Architecture

80

NKK-HUST

Bộ giải mã (Decoder)

- N đầu vào, 2^N đầu ra
- Chỉ có một đầu ra tích cực (được chọn) tương ứng với một tổ hợp của N đầu vào.

A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Jan2014 Computer Architecture 81

NKK-HUST

Thực hiện bộ giải mã 3 ra 8

Jan2014 Computer Architecture 82

NKK-HUST

Bộ cộng (Adder)

- Bộ cộng bán phần (Half-adder)
 - Cộng hai bit tạo ra bit tổng và bit nhớ
- Bộ cộng toàn phần (Full-adder)
 - Cộng 3 bit
 - Cho phép xây dựng bộ cộng N-bit

Jan2014 Computer Architecture 83

NKK-HUST

Bộ cộng (tiếp)

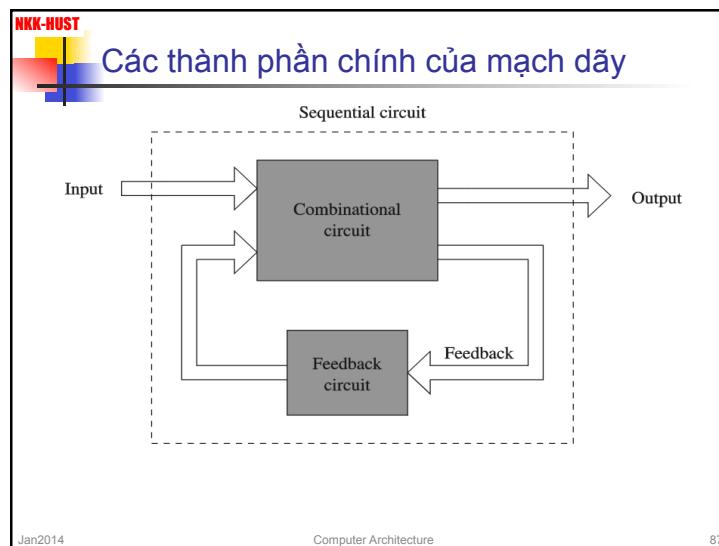
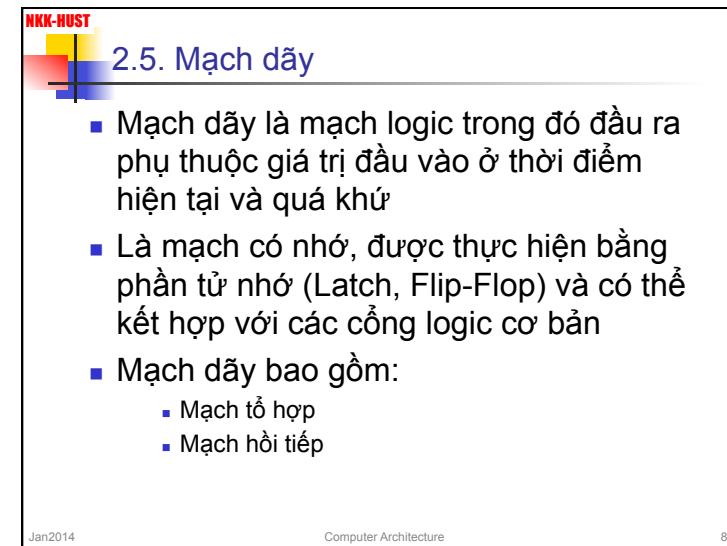
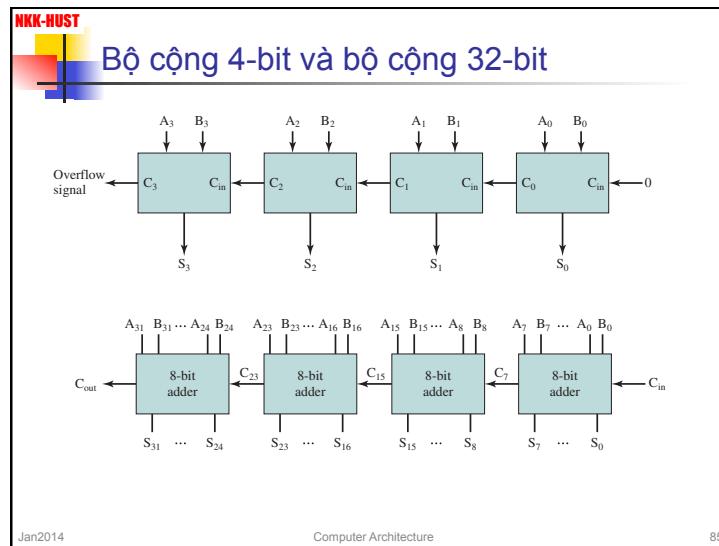
(a) Half-adder truth table and implementation

A	B	Sum	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(b) Full-adder truth table and implementation

A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Jan2014 Computer Architecture 84

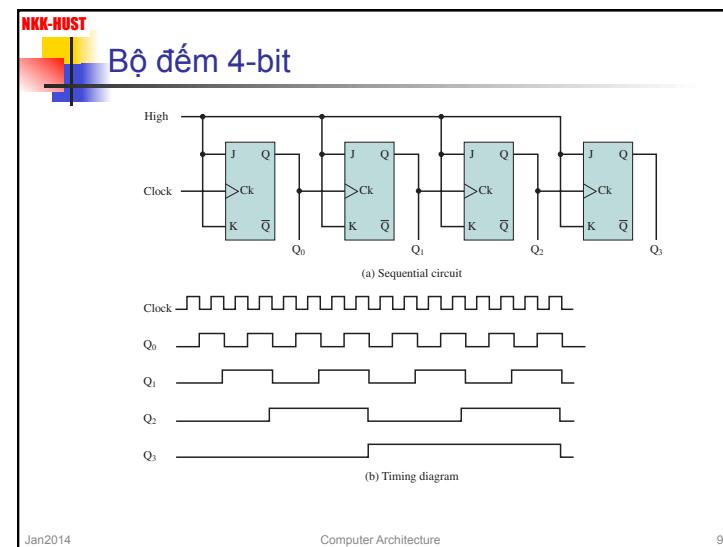
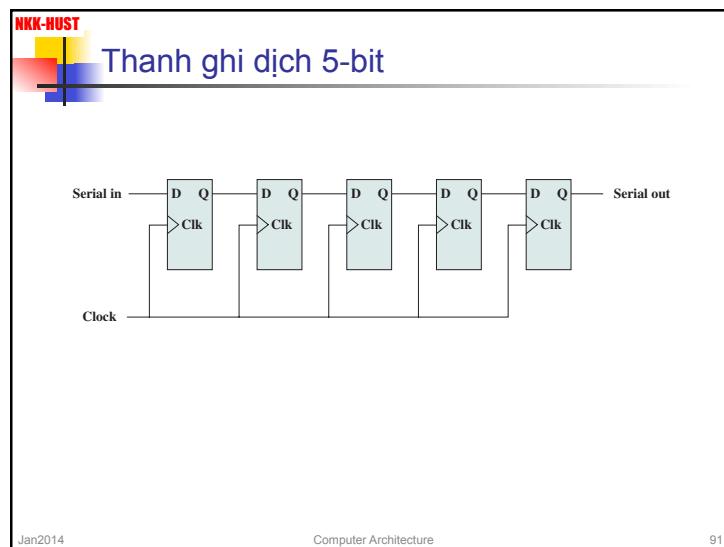
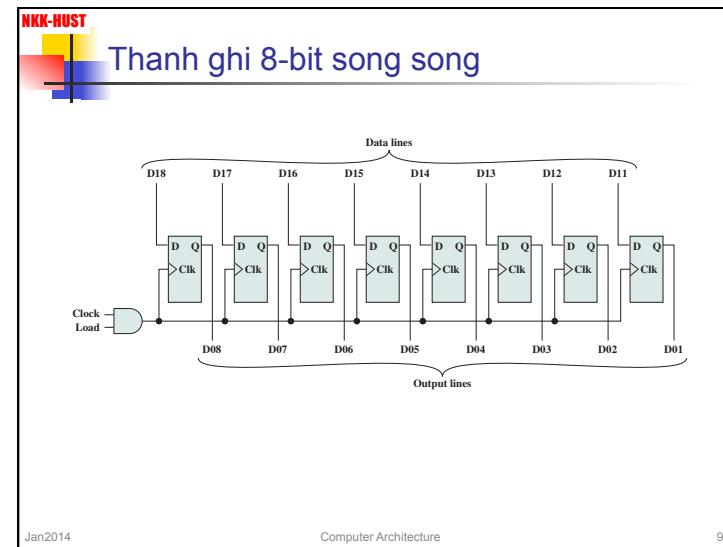
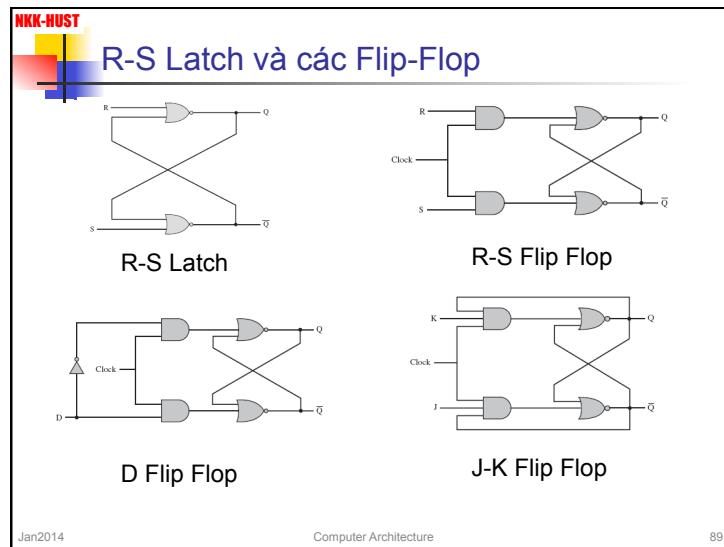


NKK-HUST

Các Flip-Flop cơ bản

Name	Graphical Symbol	Truth Table															
S-R		<table border="1"> <thead> <tr> <th>S</th> <th>R</th> <th>Q_{n+1}</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Q_n</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>-</td> </tr> </tbody> </table>	S	R	Q_{n+1}	0	0	Q_n	0	1	0	1	0	1	1	1	-
S	R	Q_{n+1}															
0	0	Q_n															
0	1	0															
1	0	1															
1	1	-															
J-K		<table border="1"> <thead> <tr> <th>J</th> <th>K</th> <th>Q_{n+1}</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Q_n</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>$\frac{1}{Q_n}$</td> </tr> <tr> <td>1</td> <td>1</td> <td>\bar{Q}_n</td> </tr> </tbody> </table>	J	K	Q_{n+1}	0	0	Q_n	0	1	0	1	0	$\frac{1}{Q_n}$	1	1	\bar{Q}_n
J	K	Q_{n+1}															
0	0	Q_n															
0	1	0															
1	0	$\frac{1}{Q_n}$															
1	1	\bar{Q}_n															
D		<table border="1"> <thead> <tr> <th>D</th> <th>Q_{n+1}</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> </tr> </tbody> </table>	D	Q_{n+1}	0	0	1	1									
D	Q_{n+1}																
0	0																
1	1																

Jan2014 Computer Architecture 88





Hết chương 2

Jan2014 Computer Architecture 93



Kiến trúc máy tính

Chương 3 HỆ THỐNG MÁY TÍNH

Nguyễn Kim Khánh
Trường Đại học Bách khoa Hà Nội

Jan2014 Computer Architecture 94



Nội dung học phần

- Chương 1. Giới thiệu chung
- Chương 2. Cơ bản về logic số
- Chương 3. Hệ thống máy tính**
- Chương 4. Số học máy tính
- Chương 5. Kiến trúc tập lệnh
- Chương 6. Bộ xử lý
- Chương 7. Bộ nhớ máy tính
- Chương 8. Hệ thống vào-ra
- Chương 9. Các kiến trúc song song

Jan2014 Computer Architecture 95



Nội dung của chương 3

- 3.1. Các thành phần cơ bản của máy tính
- 3.2. Hoạt động cơ bản của máy tính
- 3.3. Bus máy tính

Jan2014 Computer Architecture 96

NKK-HUST

3.1. Các thành phần cơ bản của máy tính

- Bộ xử lý trung tâm (CPU)
- Bộ nhớ (Memory)
- Hệ thống vào-ra (Input/Output System)
- Bus liên kết hệ thống (System Interconnection Bus)

Jan2014 Computer Architecture 97

NKK-HUST

1. Bộ xử lý trung tâm (CPU)

- Chức năng:
 - điều khiển hoạt động của máy tính
 - xử lý dữ liệu
- Nguyên tắc hoạt động cơ bản:

CPU hoạt động theo chương trình nằm trong bộ nhớ chính.

Jan2014 Computer Architecture 98

NKK-HUST

Cấu trúc cơ bản của CPU

```

graph TD
    CU[Đơn vị điều khiển (CU)] --- bus[bus bên trong]
    ALU[Đơn vị số học và logic (ALU)] --- bus
    RF[Tập các thanh ghi (RF)] --- bus
    BIU[Đơn vị nối ghép bus (BIU)] --- bus
    bus --- bus_out[bus bên ngoài]
  
```

The diagram illustrates the internal structure of a CPU. It shows four main functional units connected to a central internal bus (bus bên trong): the Control Unit (CU), the Arithmetic and Logic Unit (ALU), the Register File (RF), and the Bus Interface Unit (BIU). The BIU is also connected to an external bus (bus bên ngoài).

Jan2014 Computer Architecture 99

NKK-HUST

Các thành phần cơ bản của CPU

- **Đơn vị điều khiển (Control Unit - CU):** điều khiển hoạt động của máy tính theo chương trình đã định sẵn.
- **Đơn vị số học và logic (Arithmetic and Logic Unit - ALU):** thực hiện các phép toán số học và phép toán logic.
- **Tập thanh ghi (Register File - RF):** lưu giữ các thông tin tạm thời phục vụ cho hoạt động của CPU.
- **Đơn vị nối ghép bus (Bus Interface Unit - BIU)** kết nối và trao đổi thông tin giữa bus bên trong (*internal bus*) và bus bên ngoài (*external bus*).

Jan2014 Computer Architecture 100

NKK-HUST

2. Bộ nhớ máy tính

- Chức năng: lưu trữ chương trình và dữ liệu.
- Các thao tác cơ bản với bộ nhớ:
 - Thao tác ghi (Write)
 - Thao tác đọc (Read)
- Các thành phần chính:
 - Bộ nhớ trong (Internal Memory)
 - Bộ nhớ ngoài (External Memory)

Jan2014 Computer Architecture 101

NKK-HUST

Các thành phần của bộ nhớ máy tính

```

    graph LR
      CPU[CPU] <--> IM[Bộ nhớ trong]
      IM <--> EM[Bộ nhớ ngoài]
  
```

Jan2014 Computer Architecture 102

NKK-HUST

Bộ nhớ trong

- Chức năng và đặc điểm:
 - Chứa các thông tin mà CPU có thể trao đổi trực tiếp
 - Tốc độ rất nhanh
 - Dung lượng không lớn
 - Sử dụng bộ nhớ bán dẫn: ROM và RAM
- Các loại bộ nhớ trong:
 - Bộ nhớ chính
 - Bộ nhớ cache (bộ nhớ đệm)

Jan2014 Computer Architecture 103

NKK-HUST

Bộ nhớ chính (Main Memory)

- Chứa các chương trình và dữ liệu đang được CPU sử dụng.
- Tổ chức thành các ngăn nhớ được đánh địa chỉ.
- Ngăn nhớ thường được tổ chức theo byte.
- Nội dung của ngăn nhớ có thể thay đổi, song địa chỉ vật lý của ngăn nhớ luôn cố định.

Nội dung	Địa chỉ
1011 0010	0000
1110 0010	0001
0001 1111	0010
1010 1011	0011
0000 1000	0100
1111 1111	0101
0011 1100	0110
1000 1111	0111
1111 0001	1000
0011 1101	1001
1000 1111	1010
0011 0011	1011
1100 1101	1100
0101 1010	1101
1000 1101	1110
1111 0000	1111

Jan2014 Computer Architecture 104

Bộ nhớ cache

- Bộ nhớ có tốc độ nhanh được đặt đệm giữa CPU và bộ nhớ chính nhằm tăng tốc độ CPU truy cập bộ nhớ
- Dung lượng nhỏ hơn bộ nhớ chính
- Tốc độ nhanh hơn
- Cache thường được chia thành một số mức
- Cache có thể được tích hợp trên cùng chip bộ xử lý.
- Cache có thể có hoặc không

Jan2014

Computer Architecture

105

Bộ nhớ ngoài (External Memory)

- Chức năng và đặc điểm
 - Lưu giữ tài nguyên phần mềm của máy tính
 - Được kết nối với hệ thống dưới dạng các thiết bị vào-ra
 - Dung lượng lớn
 - Tốc độ chậm
- Các loại bộ nhớ ngoài
 - Bộ nhớ từ: Ổ đĩa cứng
 - Bộ nhớ quang: đĩa CD, DVD
 - Bộ nhớ bán dẫn: Ổ nhớ flash, thẻ nhớ, ổ SSD

Jan2014

Computer Architecture

106

3. Hệ thống vào-ra

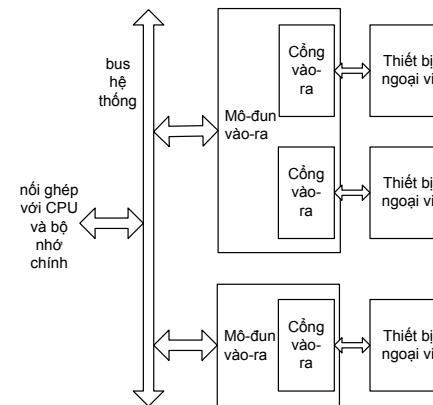
- Chức năng: Trao đổi thông tin giữa máy tính với thế giới bên ngoài.
- Các thao tác cơ bản:
 - Vào dữ liệu (Input)
 - Ra dữ liệu (Output)
- Các thành phần chính:
 - Các thiết bị ngoại vi (Peripheral Devices)
 - Các mô-đun vào-ra (IO Modules)

Jan2014

Computer Architecture

107

Cấu trúc cơ bản của hệ thống vào-ra



Jan2014

Computer Architecture

108

Các thiết bị ngoại vi

- Chức năng: chuyển đổi dữ liệu giữa bên trong và bên ngoài máy tính
- Các loại thiết bị ngoại vi cơ bản
 - Thiết bị vào: bàn phím, chuột, máy quét ...
 - Thiết bị ra: màn hình, máy in ...
 - Thiết bị nhớ: các ổ đĩa ...
 - Thiết bị truyền thông: MODEM ...

Jan2014

Computer Architecture

109

Mô-đun vào-ra

- Chức năng: nối ghép các thiết bị ngoại vi với máy tính
- Mỗi mô-đun vào-ra có một hoặc một vài cổng vào-ra (I/O Port).
- Mỗi cổng vào-ra được đánh một địa chỉ xác định.
- Các thiết bị ngoại vi được kết nối và trao đổi dữ liệu với máy tính thông qua các cổng vào-ra.

Jan2014

Computer Architecture

110

3.2. Hoạt động cơ bản của máy tính

- Thực hiện chương trình
- Hoạt động ngắt
- Hoạt động vào-ra

Jan2014

Computer Architecture

111

1. Thực hiện chương trình

- Là hoạt động cơ bản của máy tính
- Máy tính lặp đi lặp lại hai bước:
 - Nhận lệnh
 - Thực hiện lệnh
- Thực hiện chương trình bị dừng nếu thực hiện lệnh bị lỗi hoặc gặp lệnh dừng.

Jan2014

Computer Architecture

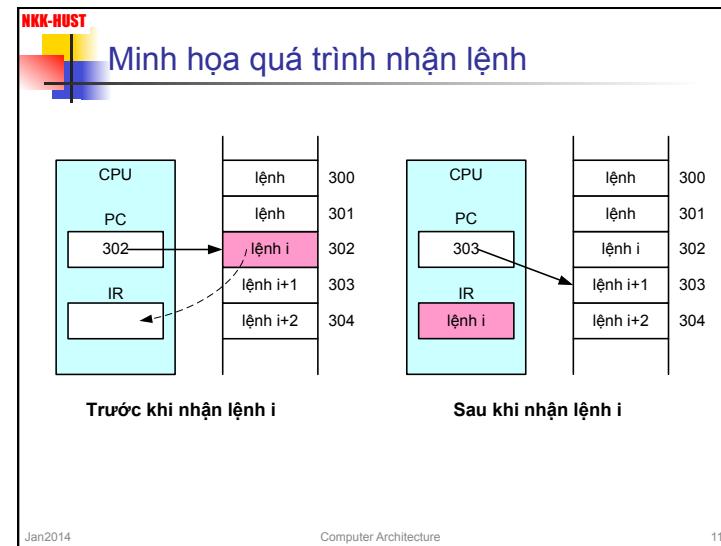
112

NKK-HUST

Nhận lệnh

- Bắt đầu mỗi chu trình lệnh, CPU nhận lệnh từ bộ nhớ chính.
- Bộ đếm chương trình PC** (Program Counter) của CPU giữ địa chỉ của lệnh sẽ được nhận.
- CPU nhận lệnh từ ngăn nhớ được trỏ bởi PC.
- Lệnh được nạp vào **thanh ghi lệnh IR** (Instruction Register).
- Sau khi lệnh được nhận vào, nội dung PC tự động tăng để trỏ sang lệnh kế tiếp.

Jan2014 Computer Architecture 113



NKK-HUST

Thực hiện lệnh

- Bộ xử lý giải mã lệnh đã được nhận và phát tín hiệu điều khiển thực hiện thao tác mà lệnh yêu cầu.
- Các kiểu thao tác của lệnh:
 - Trao đổi dữ liệu giữa CPU và bộ nhớ chính
 - Trao đổi dữ liệu giữa CPU và mô-đun vào-ra
 - Xử lý dữ liệu: thực hiện các phép toán số học hoặc phép toán logic với các dữ liệu.
 - Điều khiển rẽ nhánh
 - Kết hợp các thao tác trên.

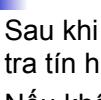
Jan2014 Computer Architecture 115

NKK-HUST

2. Hoạt động ngắt (Interrupt)

- Khái niệm chung về ngắt:** Ngắt là cơ chế cho phép CPU tạm dừng chương trình đang thực hiện để chuyển sang thực hiện một chương trình khác, gọi là **chương trình con phục vụ ngắt**.
- Các loại ngắt:**
 - Ngắt do lỗi khi thực hiện chương trình, ví dụ: tràn số, chia cho 0.
 - Ngắt do lỗi phần cứng, ví dụ lỗi bộ nhớ RAM.
 - Ngắt do mô-đun vào-ra phát tín hiệu ngắt đến CPU yêu cầu trao đổi dữ liệu.**
 - Ngắt do bộ định thời trong chế độ đa chương trình

Jan2014 Computer Architecture 116



NKK-HUST

Hoạt động ngắt (tiếp)

- Sau khi hoàn thành mỗi một lệnh, bộ xử lý kiểm tra tín hiệu ngắt
- Nếu không có ngắt → bộ xử lý nhận lệnh tiếp theo của chương trình hiện tại
- Nếu có tín hiệu ngắt:
 - Tạm dừng chương trình đang thực hiện
 - Cắt ngữ cảnh (các thông tin liên quan đến chương trình bị ngắt)
 - Thiết lập PC trả đến chương trình con phục vụ ngắt
 - Chuyển sang thực hiện chương trình con phục vụ ngắt
 - Cuối chương trình con phục vụ ngắt, khôi phục ngữ cảnh và tiếp tục chương trình đang bị tạm dừng

The diagram illustrates a context switch between two programs:

- Chương trình đang thực hiện** (Running Program):
 - A vertical stack of instruction boxes.
 - An arrow labeled "Ngắt ở đây" (Interrupt here) points to the box labeled "lệnh i".
 - The stack continues below "lệnh i" with "lệnh i+1", followed by ellipses (...), and then another "lệnh".
- Chương trình con phục vụ ngắt** (Interrupt Handler Program):
 - A separate vertical stack of instruction boxes.
 - It contains boxes labeled "lệnh", "lệnh", "lệnh", ..., and "RETURN".
 - An arrow points from the "RETURN" box back to the "lệnh i+1" box in the running program's stack.

- Xử lý với nhiều tín hiệu yêu cầu ngắt
 - Xử lý ngắt tuần tự
 - Khi một ngắt đang được thực hiện, các ngắt khác sẽ bị cấm.
 - Bộ xử lý sẽ bỏ qua các ngắt tiếp theo trong khi đang xử lý một ngắt
 - Các yêu cầu ngắt vẫn đang đợi và được kiểm tra sau khi ngắt đầu tiên được xử lý xong
 - Các ngắt được thực hiện tuần tự

NKK-HUST

Xử lý với nhiều tín hiệu yêu cầu ngắt...

- Xử lý ngắt ưu tiên
 - Các ngắt được định nghĩa mức ưu tiên khác nhau
 - Ngắt có mức ưu tiên thấp hơn có thể bị ngắt bởi ngắt ưu tiên cao hơn
 - Xảy ra ngắt lồng nhau

The diagram illustrates nested interrupt handling. It shows a vertical stack of bars representing a User Program. Two horizontal dashed lines intersect this stack, creating nested regions. The uppermost region is labeled "User Program". Below it is a region labeled "Interrupt Handler X", which contains a smaller region labeled "Interrupt Handler Y". Arrows point from the User Program to both Handler X and Handler Y, indicating that Handler X can be interrupted by Handler Y.

Jan2014

Computer Architecture

12

NKK-HUST

3. Hoạt động vào-ra

- **Hoạt động vào-ra:** là hoạt động trao đổi dữ liệu giữa mô-đun vào-ra với bên trong máy tính.
- **Các kiểu hoạt động vào-ra:**
 - CPU trao đổi dữ liệu với mô-đun vào-ra
 - Mô-đun vào-ra trao đổi dữ liệu trực tiếp với bộ nhớ chính (DMA- Direct Memory Access).

Jan2014 Computer Architecture 121

NKK-HUST

3.3. Bus máy tính

1. Luồng thông tin trong máy tính

- Các mô-đun trong máy tính:
 - CPU
 - Mô-đun nhớ
 - Mô-đun vào-ra
- ➔ cần được kết nối với nhau

Jan2014 Computer Architecture 122

NKK-HUST

Kết nối mô-đun nhớ

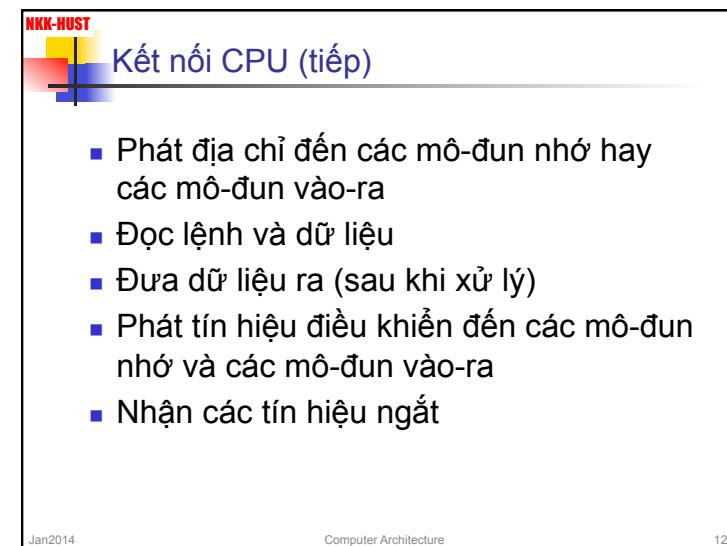
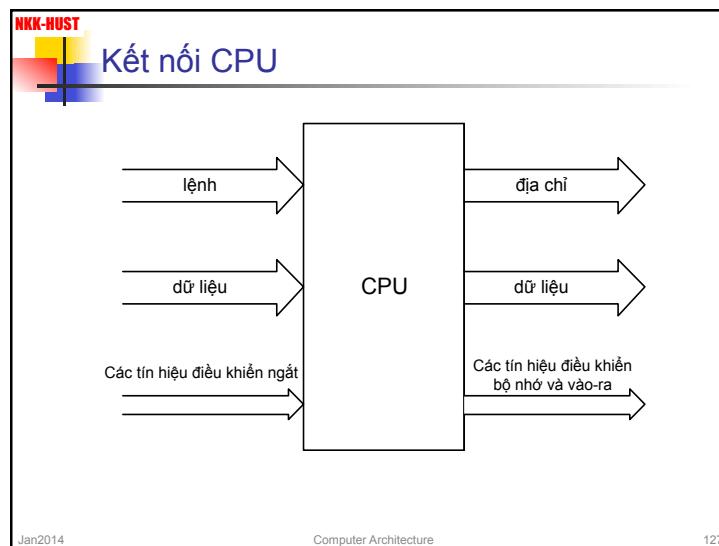
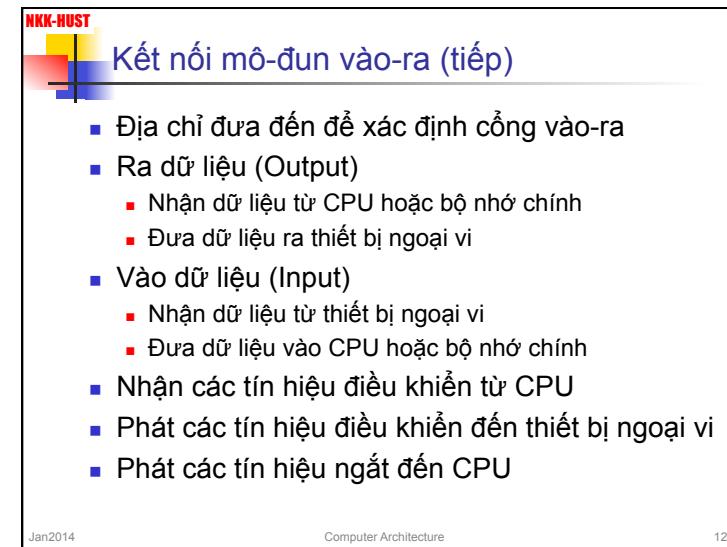
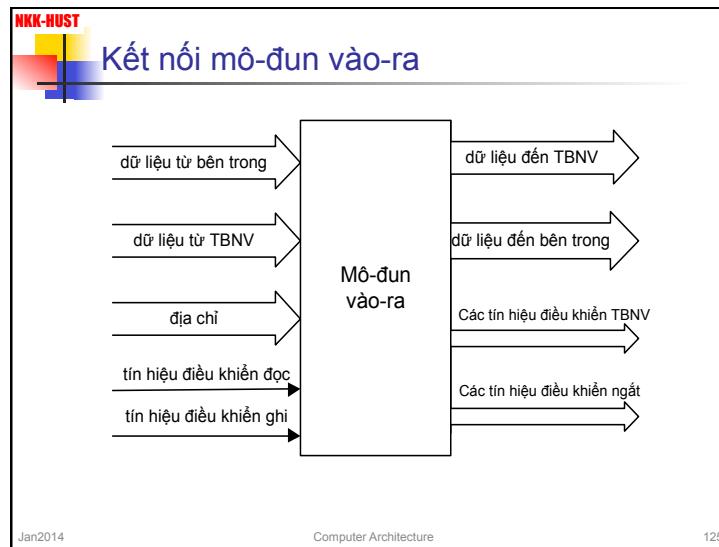
Jan2014 Computer Architecture 123

NKK-HUST

Kết nối mô-đun nhớ (tiếp)

- Địa chỉ đưa đến để xác định ngăn nhớ
- Dữ liệu được đưa đến khi ghi
- Dữ liệu hoặc lệnh được đưa ra khi đọc (lưu ý: bộ nhớ không phân biệt lệnh và dữ liệu)
- Nhận các tín hiệu điều khiển:
 - Điều khiển đọc (Read)
 - Điều khiển ghi (Write)

Jan2014 Computer Architecture 124



NKK-HUST

2. Cấu trúc bus cơ bản

- **Bus:** tập hợp các đường kết nối dùng để vận chuyển thông tin giữa các mô-đun của máy tính với nhau.
- **Các bus chức năng:**
 - Bus địa chỉ
 - Bus dữ liệu
 - Bus điều khiển
- **Độ rộng bus:** là số đường dây của bus có thể truyền các bit thông tin đồng thời (chỉ dùng cho bus địa chỉ và bus dữ liệu)

Jan2014 Computer Architecture 129

NKK-HUST

Sơ đồ cấu trúc bus cơ bản

The diagram illustrates a basic bus architecture. At the top, there are five rectangular boxes representing components: CPU, two Memory modules (labeled 'Mô-đun nhớ'), and three Input/Output modules (labeled 'Mô-đun vào-ra'). Each component is connected to three horizontal lines representing buses. The bottom-most bus is labeled 'bus điều khiển' (Control Bus). The middle bus is labeled 'bus dữ liệu' (Data Bus). The top bus is labeled 'bus địa chỉ' (Address Bus).

Jan2014 Computer Architecture 130

NKK-HUST

Bus địa chỉ

- **Chức năng:** vận chuyển địa chỉ để xác định ngăn nhớ hay cổng vào-ra
- **Độ rộng bus địa chỉ:** cho biết số lượng ngăn nhớ tối đa được đánh địa chỉ.
 - N bit: $A_{N-1}, A_{N-2}, \dots, A_2, A_1, A_0$
 - ➔ có thể đánh địa chỉ tối đa cho 2^N ngăn nhớ (không gian địa chỉ bộ nhớ)
- **Ví dụ:**
 - Bộ xử lý Pentium có bus địa chỉ 32 bit
 - ➔ có khả năng đánh địa chỉ cho 2^{32} bytes nhớ (4GBytes) (ngăn nhớ tổ chức theo byte)

Jan2014 Computer Architecture 131

NKK-HUST

Bus dữ liệu

- **Chức năng:**
 - vận chuyển lệnh từ bộ nhớ đến CPU
 - vận chuyển dữ liệu giữa CPU, bộ nhớ, mô-đun vào-ra với nhau
- **Độ rộng bus dữ liệu:** Xác định số bit dữ liệu có thể được trao đổi đồng thời.
 - M bit: $D_{M-1}, D_{M-2}, \dots, D_2, D_1, D_0$
 - M thường là 8, 16, 32, 64, 128 bit.
- **Ví dụ:** Các bộ xử lý Pentium có bus dữ liệu 64 bit

Jan2014 Computer Architecture 132

Bus điều khiển

- Chức năng: vận chuyển các tín hiệu điều khiển
- Các loại tín hiệu điều khiển:
 - Các tín hiệu điều khiển đọc/ghi
 - Các tín hiệu điều khiển ngắt
 - Các tín hiệu điều khiển bus

Jan2014

Computer Architecture

133

Một số tín hiệu điều khiển điển hình

- Các tín hiệu (phát ra từ CPU) điều khiển đọc-ghi:
 - *Memory Read (MEMR)*: điều khiển đọc dữ liệu từ một ngăn nhớ có địa chỉ xác định lên bus dữ liệu.
 - *Memory Write (MEMW)*: điều khiển ghi dữ liệu có sẵn trên bus dữ liệu đến một ngăn nhớ có địa chỉ xác định.
 - *I/O Read (IOR)*: điều khiển đọc dữ liệu từ một cổng vào-ra có địa chỉ xác định lên bus dữ liệu.
 - *I/O Write (IOW)*: điều khiển ghi dữ liệu có sẵn trên bus dữ liệu ra một cổng có địa chỉ xác định.

Jan2014

Computer Architecture

134

Một số tín hiệu điều khiển điển hình (tiếp)

- Các tín hiệu điều khiển ngắt:
 - *Interrupt Request (INTR)*: Tín hiệu từ bộ điều khiển vào-ra gửi đến yêu cầu ngắt CPU để trao đổi vào-ra. Tín hiệu INTR có thể bị che.
 - *Interrupt Acknowledge (INTA)*: Tín hiệu phát ra từ CPU báo cho bộ điều khiển vào-ra biết CPU chấp nhận ngắt để trao đổi vào-ra.
 - *Non Maskable Interrupt (NMI)*: tín hiệu ngắt không che được gửi đến ngắt CPU.
 - *Reset*: Tín hiệu từ bên ngoài gửi đến CPU và các thành phần khác để khởi động lại máy tính.

Jan2014

Computer Architecture

135

Một số tín hiệu điều khiển điển hình (tiếp)

- Các tín hiệu điều khiển bus:
 - *Bus Request (BRQ)* hay là *Hold*: Tín hiệu từ mô-đun điều khiển vào-ra gửi đến yêu cầu CPU chuyển nhượng quyền sử dụng bus.
 - *Bus Grant (BGT)* hay là *Hold Acknowledge (HLDA)*: Tín hiệu phát ra từ CPU chấp nhận chuyển nhượng quyền sử dụng bus.
 - *Lock/ Unlock*: Tín hiệu *cấm/cho-phép* xin chuyển nhượng bus

Jan2014

Computer Architecture

136

Đặc điểm của cấu trúc đơn bus

- Bus hệ thống chỉ phục vụ được một yêu cầu trao đổi dữ liệu tại một thời điểm
- Bus hệ thống phải có tốc độ bằng tốc độ bus của mô-đun nhanh nhất trong hệ thống
- Bus hệ thống phụ thuộc vào cấu trúc bus (các tín hiệu) của bộ xử lý → các mô-đun nhớ và các mô-đun vào-ra cũng phụ thuộc vào bộ xử lý.
- Khắc phục: phân cấp bus → cấu trúc đa bus

Jan2014

Computer Architecture

137

3. Phân cấp bus trong máy tính

- Tổ chức thành nhiều bus trong hệ thống máy tính
 - Cho các thành phần khác nhau:
 - Bus của bộ xử lý
 - Bus của bộ nhớ chính
 - Các bus vào-ra
 - Các bus khác nhau về tốc độ
 - Bus bộ nhớ chính và các bus vào-ra không phụ thuộc vào bộ xử lý cụ thể.

Jan2014

Computer Architecture

138

Một số bus điển hình trong máy tính

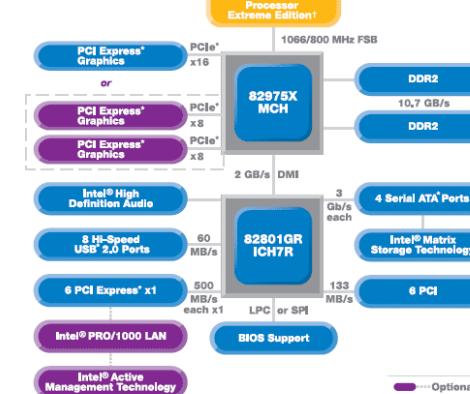
- Bus của bộ xử lý: có tốc độ nhanh nhất
- Bus của bộ nhớ chính (nối ghép với các mô-đun RAM)
- PCI Express bus (Peripheral Component Interconnect): nối ghép với các thiết bị ngoại vi có tốc độ trao đổi dữ liệu nhanh.
- SATA (Serial Advanced Technology Attachment): Bus kết nối với ổ đĩa cứng hoặc ổ đĩa CD/DVD
- USB (Universal Serial Bus): Bus nối tiếp đa năng

Jan2014

Computer Architecture

139

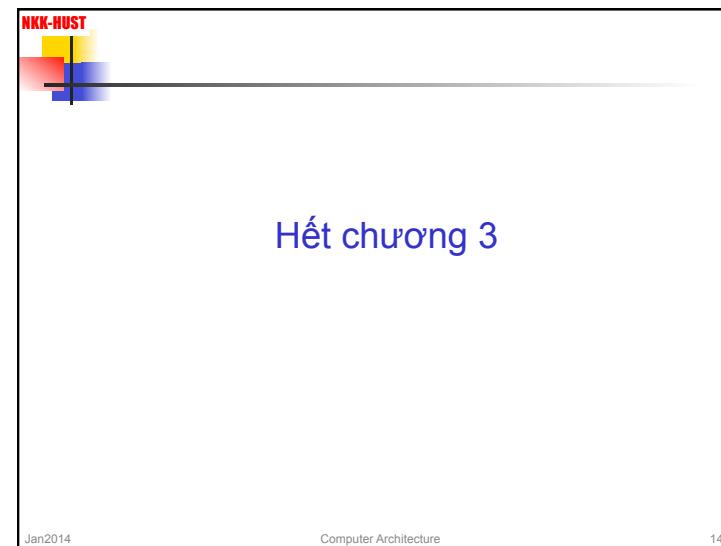
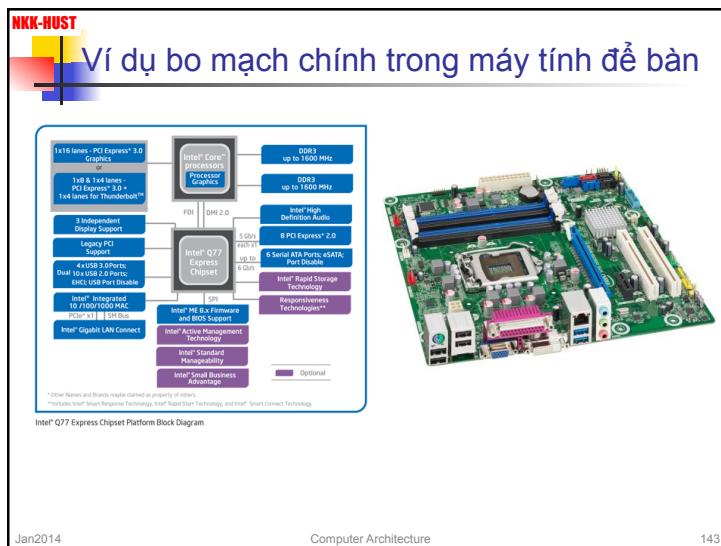
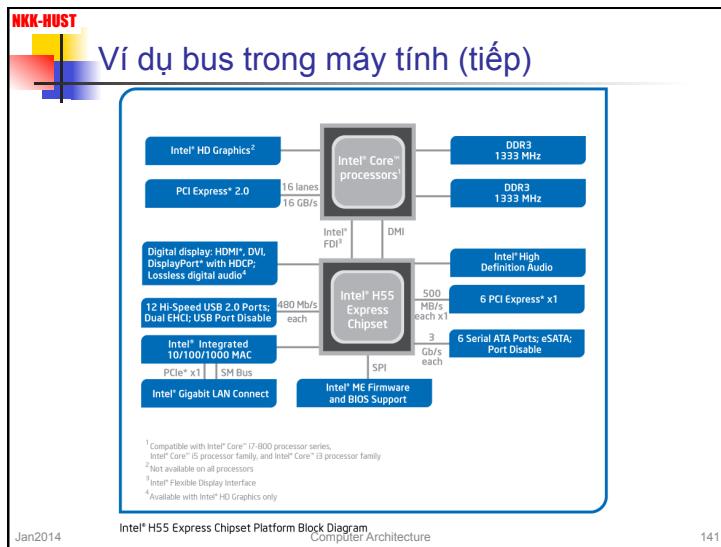
Ví dụ bus trong máy tính



Jan2014

Computer Architecture

140



NKK-HUST

Kiến trúc máy tính

Chương 4 SỐ HỌC MÁY TÍNH

Nguyễn Kim Khánh
Trường Đại học Bách khoa Hà Nội

Jan2014 Computer Architecture 145

NKK-HUST

Nội dung học phần

- Chương 1. Giới thiệu chung
- Chương 2. Cơ bản về logic số
- Chương 3. Hệ thống máy tính
- Chương 4. Số học máy tính**
- Chương 5. Kiến trúc tập lệnh
- Chương 6. Bộ xử lý
- Chương 7. Bộ nhớ máy tính
- Chương 8. Hệ thống vào-ra
- Chương 9. Các kiến trúc song song

Jan2014 Computer Architecture 146

NKK-HUST

Nội dung chương 4

- 4.1. Biểu diễn số nguyên
- 4.2. Phép cộng và phép trừ số nguyên
- 4.3. Phép nhân và phép chia số nguyên
- 4.4. Số dấu phẩy động

Jan2014 Computer Architecture 147

NKK-HUST

4.1. Biểu diễn số nguyên

- Số nguyên không dấu (Unsigned Integer)
- Số nguyên có dấu (Signed Integer)

Jan2014 Computer Architecture 148

NKK-HUST 1. Biểu diễn số nguyên không dấu

- Nguyên tắc tổng quát: Dùng n bit biểu diễn số nguyên không dấu A:

$$a_{n-1}a_{n-2} \dots a_2a_1a_0$$

Giá trị của A được tính như sau:

$$A = \sum_{i=0}^{n-1} a_i 2^i$$

Dải biểu diễn của A: từ 0 đến $2^n - 1$

Jan2014

Computer Architecture

149

NKK-HUST Các ví dụ

- Ví dụ 1. Biểu diễn các số nguyên không dấu sau đây bằng 8-bit:

$$A = 41 ; B = 150$$

Giải:

$$A = 41 = 32 + 8 + 1 = 2^5 + 2^3 + 2^0$$

$$41 = 0010\ 1001$$

$$B = 150 = 128 + 16 + 4 + 2 = 2^7 + 2^4 + 2^2 + 2^1$$

$$150 = 1001\ 0110$$

Jan2014

Computer Architecture

150

NKK-HUST Các ví dụ (tiếp)

- Ví dụ 2. Cho các số nguyên không dấu M, N được biểu diễn bằng 8-bit như sau:

- M = 0001 0010
- N = 1011 1001

Xác định giá trị của chúng ?

Giải:

- M = 0001 0010 = $2^4 + 2^1 = 16 + 2 = 18$
- N = 1011 1001 = $2^7 + 2^5 + 2^4 + 2^3 + 2^0$
 $= 128 + 32 + 16 + 8 + 1 = 185$

Jan2014

Computer Architecture

151

NKK-HUST Với n = 8 bit

Biểu diễn được các giá trị từ 0 đến 255

0000 0000	= 0	Chú ý:
0000 0001	= 1	1111 1111
0000 0010	= 2	+ <u>0000 0001</u>
0000 0011	= 3	1 0000 0000
...		
1111 1111	= 255	Vậy: $255 + 1 = 0$? \rightarrow do tràn nhớ ra ngoài

Jan2014

Computer Architecture

152

NKK-HUST

Trục số học với $n = 8$ bit

Trục số học:

Trục số học máy tính:

Jan2014 Computer Architecture 153

NKK-HUST

Với $n = 16$ bit, 32 bit, 64 bit

- $n = 16$ bit: dải biểu diễn từ 0 đến $65535 (2^{16} - 1)$
 - 0000 0000 0000 0000 = 0
 - ...
 - 0000 0000 1111 1111 = 255
 - 0000 0001 0000 0000 = 256
 - ...
 - 1111 1111 1111 1111 = 65535
- $n = 32$ bit: dải biểu diễn từ 0 đến $2^{32} - 1$
- $n = 64$ bit: dải biểu diễn từ 0 đến $2^{64} - 1$

Jan2014 Computer Architecture 154

NKK-HUST

2. Biểu diễn số nguyên có dấu

Số bù chín và Số bù mươi

- Cho một số thập phân A được biểu diễn bằng n chữ số thập phân, ta có:
 - Số bù chín của A = $(10^n - 1) - A$
 - Số bù mươi của A = $10^n - A$
- Số bù mươi của A = (Số bù chín của A) + 1

Jan2014 Computer Architecture 155

NKK-HUST

Số bù chín và Số bù mươi (tiếp)

- Ví dụ: với $n=4$, cho A = 3265
 - Số bù chín của A:

$$\begin{array}{r} 9999 & (10^4 - 1) \\ - 3265 & (A) \\ \hline 6734 \end{array}$$
 - Số bù mươi của A:

$$\begin{array}{r} 10000 & (10^4) \\ - 3265 & (A) \\ \hline 6735 \end{array}$$

Jan2014 Computer Architecture 156

NKK-HUST

Số bù một và Số bù hai

- Định nghĩa: Cho một số nhị phân A được biểu diễn bằng n bit, ta có:
 - Số bù một của A = $(2^n - 1) - A$
 - Số bù hai của A = $2^n - A$
- Số bù hai của A = (Số bù một của A) + 1

Jan2014

Computer Architecture

157

NKK-HUST

Số bù một và Số bù hai (tiếp)

Ví dụ: với n = 8 bit, cho A = 0010 0101

- Số bù một của A được tính như sau:

$$\begin{array}{r} 1111\ 1111 \quad (2^8 - 1) \\ - 0010\ 0101 \quad (A) \\ \hline 1101\ 1010 \end{array}$$

→ đảo các bit của A
- Số bù hai của A được tính như sau:

$$\begin{array}{r} 1\ 0000\ 0000 \quad (2^8) \\ - 0010\ 0101 \quad (A) \\ \hline 1101\ 1011 \end{array}$$

→ thực hiện khó khăn

Jan2014

Computer Architecture

158

NKK-HUST

Quy tắc tìm Số bù một và Số bù hai

- Số bù một của A = đảo giá trị các bit của A
- (Số bù hai của A) = (Số bù một của A) + 1
- Ví dụ:
 - Cho A = 0010 0101
 - Số bù một = 1101 1010
 - Số bù hai = $\begin{array}{r} + 1 \\ \hline 1101\ 1011 \end{array}$
- Nhận xét:

$$\begin{array}{rcl} A & = & 0010\ 0101 \\ \text{Số bù hai} & = & + \underline{1101\ 1011} \\ & & \textcolor{red}{1}\ \textcolor{green}{0000}\ \textcolor{red}{0000} = 0 \end{array}$$

(bỏ qua bit nhớ ra ngoài)

→ Số bù hai của A = -A

Jan2014

Computer Architecture

159

NKK-HUST

Biểu diễn số nguyên có dấu bằng mã bù hai

Nguyên tắc tổng quát: Dùng n bit biểu diễn số nguyên có dấu A:

$$a_{n-1}a_{n-2} \dots a_2a_1a_0$$

- **Với A là số dương:** bit $a_{n-1} = 0$, các bit còn lại biểu diễn độ lớn như số không dấu
- **Với A là số âm:** được biểu diễn bằng số bù hai của số dương tương ứng, vì vậy bit $a_{n-1} = 1$

Jan2014

Computer Architecture

160

NKK-HUST

Biểu diễn số dương

- Dạng tổng quát của số dương A:
$$0a_{n-2} \dots a_2 a_1 a_0$$
- Giá trị của số dương A:
$$A = \sum_{i=0}^{n-2} a_i 2^i$$
- Dải biểu diễn cho số dương: 0 đến $2^{n-1}-1$

Jan2014 Computer Architecture 161

NKK-HUST

Biểu diễn số âm

- Dạng tổng quát của số âm A:
$$1a_{n-2} \dots a_2 a_1 a_0$$
- Giá trị của số âm A:
$$A = -2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$$
- Dải biểu diễn cho số âm: -1 đến -2^{n-1}

Jan2014 Computer Architecture 162

NKK-HUST

Biểu diễn tổng quát cho số nguyên có dấu

- Dạng tổng quát của số nguyên A:
$$a_{n-1} a_{n-2} \dots a_2 a_1 a_0$$
- Giá trị của A được xác định như sau:
$$A = -a_{n-1} 2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$$
- Dải biểu diễn: từ $-(2^{n-1})$ đến $+(2^{n-1}-1)$

Jan2014 Computer Architecture 163

NKK-HUST

Các ví dụ

- Ví dụ 1. Biểu diễn các số nguyên có dấu sau đây bằng 8-bit:
 $A = +58 ; B = -80$
- Giải:

$A = +58$	=	$0011\ 1010$
$B = -80$	=	$1010\ 0000$
		Ta có: $+80 = 0101\ 0000$
		Số bù một = $1010\ 1111$
		Số bù hai = $\begin{array}{r} + 1 \\ \hline 1011\ 0000 \end{array}$

Vậy: $B = -80 = 1011\ 0000$

Jan2014 Computer Architecture 164

NKK-HUST

Các ví dụ

- Ví dụ 2. Hãy xác định giá trị của các số nguyên có dấu được biểu diễn dưới đây:
 - P = 0110 0010
 - Q = 1101 1011

Giải:

- P = 0110 0010 = $64+32+2 = +98$
- Q = 1101 1011 = $-128+64+16+8+2+1 = -37$

Jan2014

Computer Architecture

165

NKK-HUST

Với n = 8 bit

Biểu diễn được các giá trị từ -128 đến +127

0000 0000	=	0					
0000 0001	=	+1					
0000 0010	=	+2					
0000 0011	=	+3					
...							
0111 1111	=	+127					
1000 0000	=	-128					
1000 0001	=	-127					
...							
1111 1110	=	-2					
1111 1111	=	-1					

Chú ý:
 $+127 + 1 = -128$
 $(-128) + (-1) = +127$
 \rightarrow do tràn xảy ra

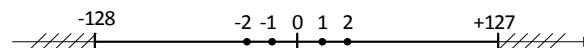
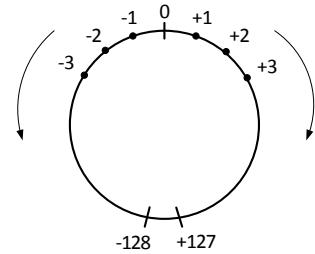
Jan2014

Computer Architecture

166

NKK-HUST

Trục số học số nguyên có dấu với n = 8 bit

- Trục số học:
- Trục số học máy tính:


Jan2014

Computer Architecture

167

NKK-HUST

Với n = 16 bit, 32 bit, 64 bit

- Với n=16bit: biểu diễn từ -32768 đến +32767
 - 0000 0000 0000 0000 = 0
 - 0000 0000 0000 0001 = +1
 - ...
 - 0111 1111 1111 1111 = +32767
 - 1000 0000 0000 0000 = -32768
 - ...
 - 1111 1111 1111 1111 = -1
- Với n=32bit: biểu diễn từ -2^{31} đến $2^{31}-1$
- Với n=64bit: biểu diễn từ -2^{63} đến $2^{63}-1$

Jan2014

Computer Architecture

168

NKK-HUST

Chuyển đổi dữ liệu từ 8 bit thành 16 bit

- Đối với số dương:
 $+19 = \begin{array}{r} 0001\ 0011 \\ (8bit) \end{array}$
 $+19 = \begin{array}{r} 0000\ 0000\ 0001\ 0011 \\ (16bit) \end{array}$
 → thêm 8 bit 0 bên trái
- Đối với số âm:
 $-19 = \begin{array}{r} 1110\ 1101 \\ (8bit) \end{array}$
 $-19 = \begin{array}{r} 1111\ 1111\ 1110\ 1101 \\ (16bit) \end{array}$
 → thêm 8 bit 1 bên trái

Jan2014 Computer Architecture 169

NKK-HUST

4.2. Thực hiện phép cộng/trừ với số nguyên

1. Phép cộng số nguyên không dấu

Bộ cộng n-bit

```

graph TD
    Y[n bit] --> Adder[ ]
    X[n bit] --> Adder[ ]
    Cin[ ] --> Adder[ ]
    Adder[ ] --> Cout[ ]
    Adder[ ] --> S[n bit]
  
```

Jan2014 Computer Architecture 170

NKK-HUST

Nguyên tắc cộng số nguyên không dấu

Khi cộng hai số nguyên không dấu n-bit, kết quả nhận được là n-bit:

- Nếu $C_{out}=0$ → nhận được kết quả đúng.
- Nếu $C_{out}=1$ → nhận được kết quả sai, do tràn nhớ ra ngoài (Carry Out).
- Tràn nhớ ra ngoài khi: $\text{tổng} > (2^n - 1)$

Jan2014 Computer Architecture 171

NKK-HUST

Ví dụ cộng số nguyên không dấu

- $\begin{array}{r} 57 \\ + 34 \\ \hline 91 \end{array} = \begin{array}{r} 0011\ 1001 \\ + 0010\ 0010 \\ \hline 0101\ 1011 \end{array} = 64+16+8+2+1=91 \rightarrow \text{đúng}$
- $\begin{array}{r} 209 \\ + 73 \\ \hline 282 \end{array} = \begin{array}{r} 1101\ 0001 \\ + 0100\ 1001 \\ \hline 1\ 0001\ 1010 \end{array} = 16+8+2=26 \rightarrow \text{sai}$
→ có tràn nhớ ra ngoài ($C_{out}=1$)

Để có kết quả đúng ta thực hiện cộng theo 16-bit:

$$\begin{array}{r} 209 = 0000\ 0000\ 1101\ 0001 \\ + 73 = + 0000\ 0000\ 0100\ 1001 \\ \hline 0000\ 0001\ 0001\ 1010 = 256+16+8+2 = 282 \end{array}$$

Jan2014 Computer Architecture 172

2. Phép đảo dấu

- Ta có:

$$\begin{array}{rcl} + 37 & = & 0010\ 0101 \\ \text{bù một} & = & 1101\ 1010 \\ & + & \underline{\quad\quad\quad 1} \\ \text{bù hai} & = & 1101\ 1011 = -37 \end{array}$$

- Lấy bù hai của số âm:

$$\begin{array}{rcl} -37 & = & 1101\ 1011 \\ \text{bù một} & = & 0010\ 0100 \\ & + & \underline{\quad\quad\quad 1} \\ \text{bù hai} & = & 0010\ 0101 = +37 \end{array}$$

- Kết luận: *Phép đảo dấu số nguyên trong máy tính thực chất là lấy bù hai*

Jan2014

Computer Architecture

173

3. Cộng số nguyên có dấu

Khi cộng hai số nguyên có dấu n-bit, kết quả nhận được là n-bit và **không cần quan tâm đến bit C_{out}**

- Cộng hai số khác dấu: **kết quả luôn luôn đúng.**
- Cộng hai số cùng dấu:
 - nếu dấu kết quả cùng dấu với các số hạng thì **kết quả là đúng.**
 - nếu kết quả có dấu ngược lại, khi đó có **tràn xảy ra (Overflow) và kết quả bị sai.**
- Tràn xảy ra khi tổng nằm ngoài dải biểu diễn:

$$[-(2^{n-1}), + (2^{n-1}-1)]$$

Jan2014

Computer Architecture

174

Ví dụ cộng số nguyên có dấu không tràn

$$\begin{array}{rcl} (+70) & = & 0100\ 0110 \\ + (+42) & = & 0010\ 1010 \\ & + & \underline{0111\ 0000} = +112 \\ \\ (+97) & = & 0110\ 0001 \\ + (-52) & = & 1100\ 1100 \quad (+52=0011\ 0100) \\ & + & \underline{1010\ 1101} = +45 \\ \\ (-90) & = & 1010\ 0110 \quad (+90=0101\ 1010) \\ + (+36) & = & 0010\ 0100 \\ & + & \underline{1100\ 1010} = -54 \\ \\ (-74) & = & 1011\ 0110 \quad (+74=0100\ 1010) \\ + (-30) & = & 1110\ 0010 \quad (+30=0001\ 1110) \\ & + & \underline{1001\ 1000} = -104 \end{array}$$

Jan2014

Computer Architecture

175

Ví dụ cộng số nguyên có dấu bị tràn

$$\begin{array}{rcl} (+75) & = & 0100\ 1011 \\ + (+82) & = & 0101\ 0010 \\ & + & \underline{1001\ 1101} \\ & & = -128+16+8+4+1 = -99 \rightarrow \text{sai} \\ \\ (-104) & = & 1001\ 1000 \quad (+104=0110\ 1000) \\ + (-43) & = & 1101\ 0101 \quad (+43=0010\ 1011) \\ & + & \underline{-147} \quad \underline{10110\ 1101} \\ & & = 64+32+8+4+1 = +109 \rightarrow \text{sai} \\ \\ \text{Cả hai ví dụ đều tràn vì tổng nằm ngoài dải} \\ \text{biểu diễn } [-128, +127] \end{array}$$

Jan2014

Computer Architecture

176

NKK-HUST

4. Nguyên tắc thực hiện phép trừ

- Phép trừ hai số nguyên: $X - Y = X + (-Y)$
- Nguyên tắc: Lấy bù hai của Y để được $-Y$, rồi cộng với X

Jan2014 Computer Architecture 177

NKK-HUST

4.3. Phép nhân và phép chia số nguyên

1. Nhân số nguyên không dấu

$$\begin{array}{r}
 1011 & \text{Số bị nhân (11)} \\
 \times 1101 & \text{Số nhân (13)} \\
 \hline
 1011 \\
 0000 \\
 1011 \\
 \hline
 1011 \\
 \hline
 10001111 & \text{Tích (143)}
 \end{array}$$

Các tích riêng phần

Jan2014 Computer Architecture 178

NKK-HUST

Nhân số nguyên không dấu (tiếp)

- Các tích riêng phần được xác định như sau:
 - Nếu bit của số nhân bằng 0 → tích riêng phần bằng 0.
 - Nếu bit của số nhân bằng 1 → tích riêng phần bằng số bị nhân.
 - Tích riêng phần tiếp theo được dịch trái một bit so với tích riêng phần trước đó.
- Tích bằng tổng các tích riêng phần
- Nhân hai số nguyên n -bit, tích có độ dài $2n$ bit (không bao giờ tràn).

Jan2014 Computer Architecture 179

NKK-HUST

Bộ nhân số nguyên không dấu

Số bị nhân: $M_{n-1} M_{n-2} \dots M_1 M_0$

Bộ nhân: $A_{n-1} A_{n-2} \dots A_1 A_0$

Bộ công n-bit

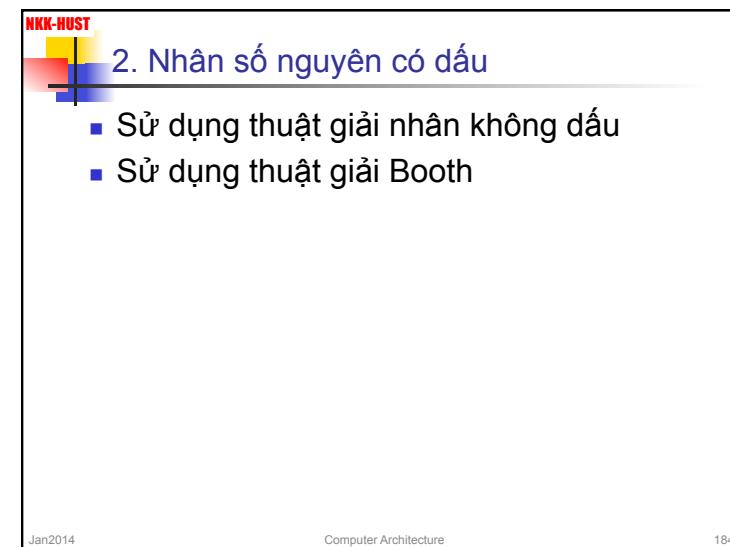
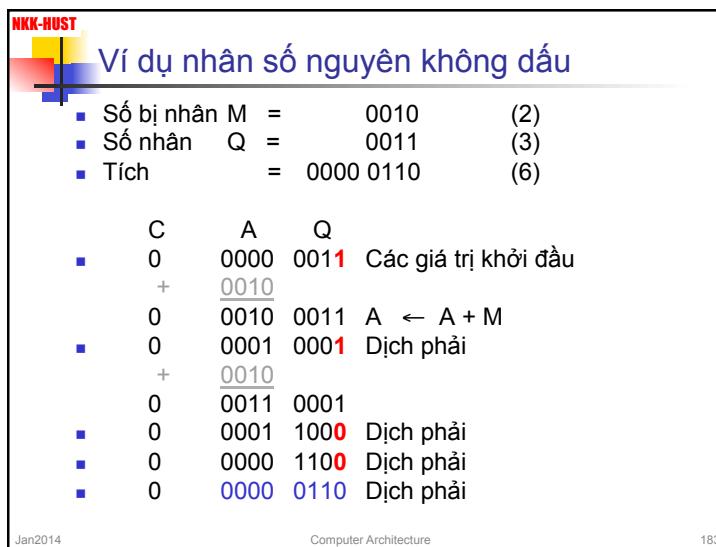
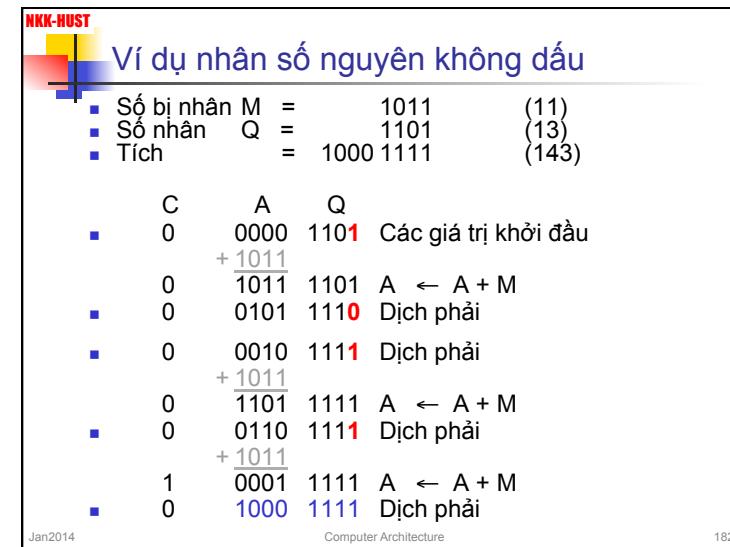
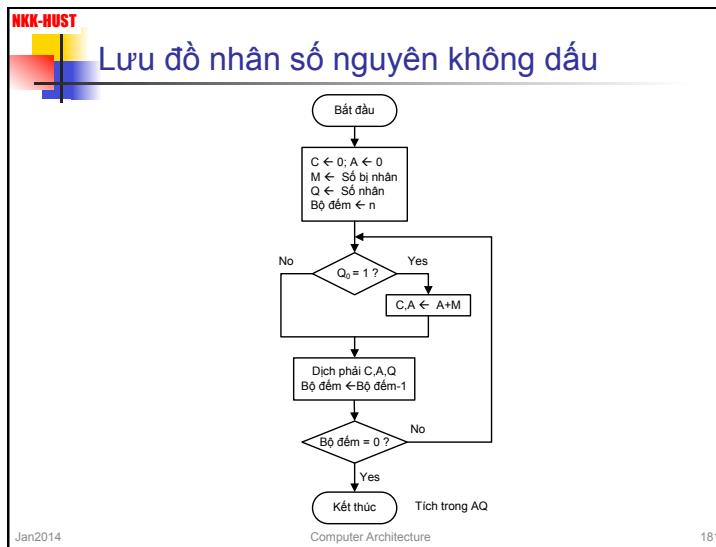
Bộ logic điều khiển cộng và dịch

Điều khiển cộng

Điều khiển dịch phải

Số nhân: $Q_{n-1} Q_{n-2} \dots Q_1 Q_0$

Jan2014 Computer Architecture 180

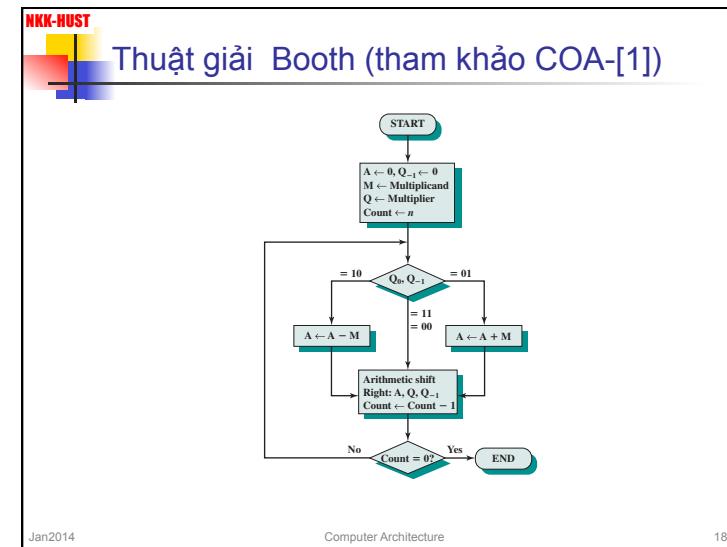


NKK-HUST

Sử dụng thuật giải nhân không dấu

- Bước 1. Chuyển đổi số bị nhân và số nhân thành số dương tương ứng
- Bước 2. Nhân hai số dương bằng thuật giải nhân số nguyên không dấu, được tích của hai số dương.
- Bước 3. Hiệu chỉnh dấu của tích:
 - Nếu hai thừa số ban đầu cùng dấu thì giữ nguyên kết quả ở bước 2.
 - Nếu hai thừa số ban đầu là khác dấu thì đảo dấu kết quả của bước 2 (lấy bù hai).

Jan2014 Computer Architecture 185



NKK-HUST

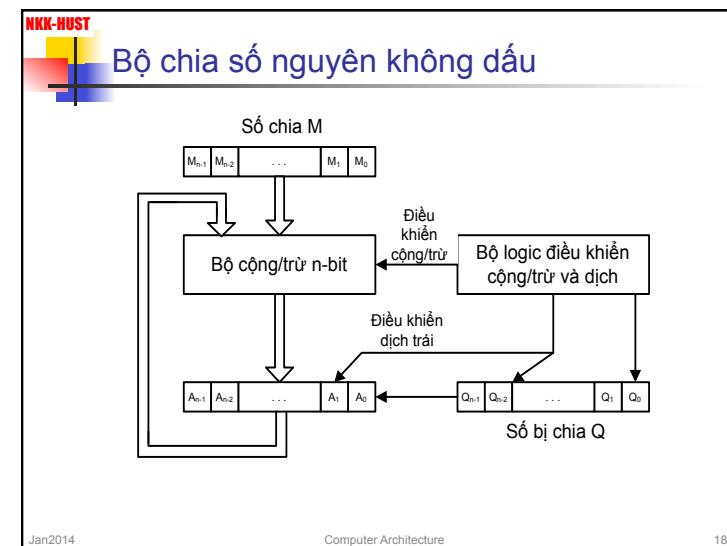
3. Chia số nguyên không dấu

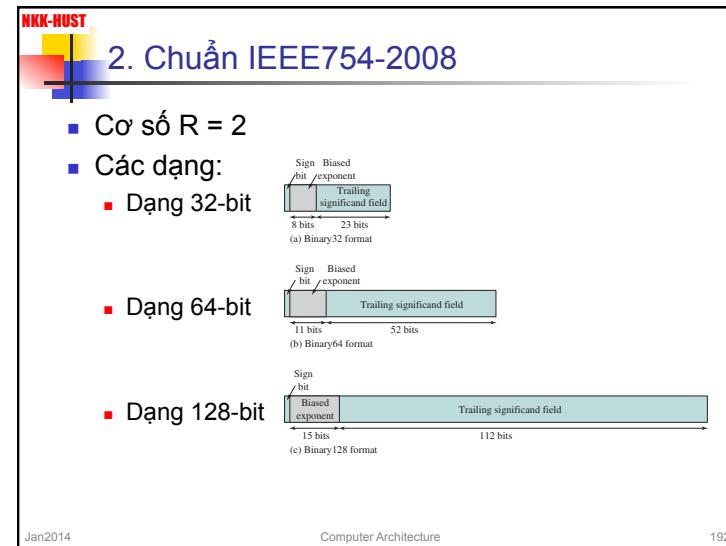
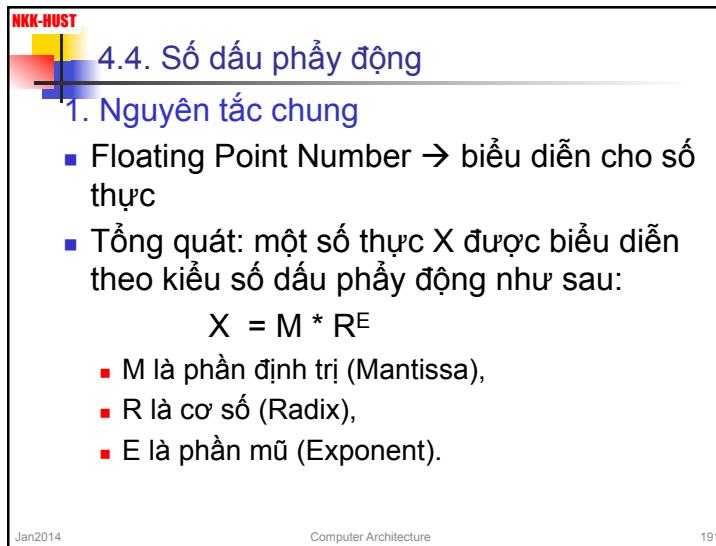
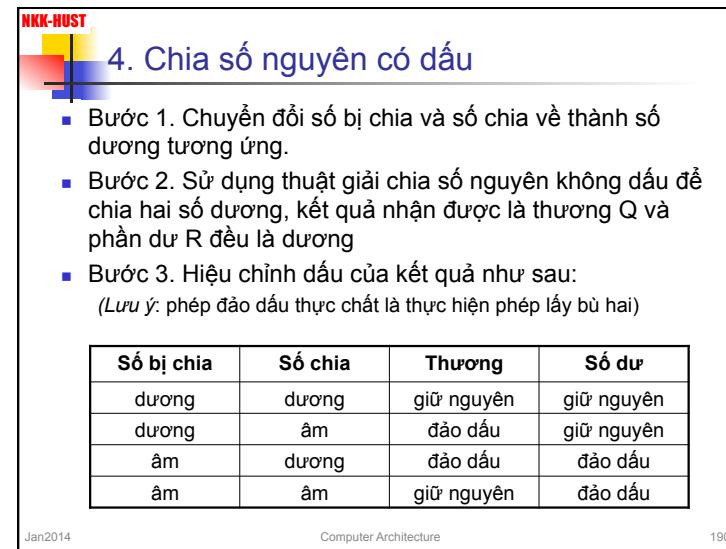
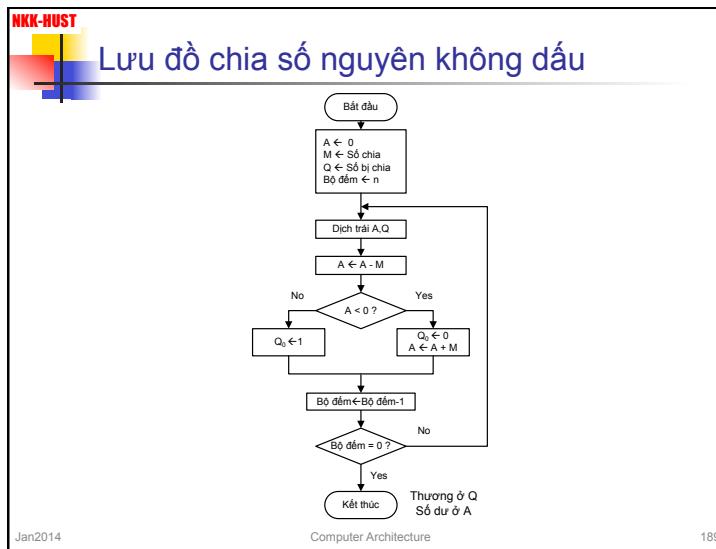
Số bị chia
$$\begin{array}{r} 10010011 \\ \underline{- 1011} \\ 001110 \end{array}$$
 Số chia Thương

$$\begin{array}{r} 1011 \\ \underline{\quad\quad} \\ 00001101 \end{array}$$

Số bị chia
$$\begin{array}{r} 001111 \\ \underline{- 1011} \\ 100 \end{array}$$
 Phần dư

Jan2014 Computer Architecture 187





Dạng 32 bit

31	30	23	22	0
S	e	m		

- S là bit dấu:
 - S = 0 → số dương
 - S = 1 → số âm
- e (8 bit) là mã *excess-127* của phần mũ E:
 - $e = E+127 \rightarrow E = e - 127$
 - giá trị 127 gọi là là độ lệch (bias)
- m (23 bit) là phần lẻ của phần định trị M:
 - $M = 1.m$
- Công thức xác định giá trị của số thực:

$$X = (-1)^S \cdot 1.m \cdot 2^{e-127}$$

Jan2014

Computer Architecture

193

Ví dụ 1

Xác định giá trị của số thực được biểu diễn bằng 32-bit như sau:

- $1100\ 0001\ 0101\ 0110\ 0000\ 0000\ 0000\ 0000$
 - S = 1 → số âm
 - e = $1000\ 0010_2 = 130 \rightarrow E = 130-127=3$
- Vậy

$$X = -1.10101100 \cdot 2^3 = -1101.011 = -13.375$$
- $0011\ 1111\ 1000\ 0000\ 0000\ 0000\ 0000\ 0000 = ?$
 - +1.0

Jan2014

Computer Architecture

194

Ví dụ 2

Biểu diễn số thực $X = 83.75$ về dạng số dấu phẩy động IEEE754 32-bit

Giải:

- $X = 83.75_{(10)} = 1010011.11_{(2)} = 1.01001111 \times 2^6$
- Ta có:
 - S = 0 vì đây là số dương
 - $E = e-127 = 6 \rightarrow e = 127 + 6 = 133_{(10)} = 1000\ 0101_{(2)}$
- Vậy:

$$X = 0100\ 0010\ 1010\ 0111\ 1000\ 0000\ 0000$$

Jan2014

Computer Architecture

195

Ví dụ 3

Biểu diễn số thực $X = -0,2$ về dạng số dấu phẩy động IEEE754 32-bit

Giải:

- $X = -0,2_{(10)} = -0.00110011...0011..._{(2)} = -1.100110011..0011... \times 2^{-3}$
- Ta có:
 - S = 1 vì đây là số âm
 - $E = e-127 = -3 \rightarrow e = 127 - 3 = 124_{(10)} = 0111\ 1100_{(2)}$
- Vậy:

$$X = 1011\ 1110\ 0100\ 1100\ 1100\ 1100\ 1100$$

Jan2014

Computer Architecture

196

Bài tập

Biểu diễn các số thực sau đây về dạng số dấu phẩy động IEEE754 32-bit:

$$X = -27.0625; \quad Y = 1/32$$

Jan2014

Computer Architecture

197

Các qui ước đặc biệt

- Các bit của e bằng 0, các bit của m bằng 0, thì $X = \pm 0$
 $x000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000 \rightarrow X = \pm 0$
- Các bit của e bằng 1, các bit của m bằng 0, thì $X = \pm \infty$
 $x111\ 1111\ 1000\ 0000\ 0000\ 0000\ 0000 \rightarrow X = \pm \infty$
- Các bit của e bằng 1, còn m có ít nhất một bit bằng 1, thì nó không biểu diễn cho số nào cả (NaN - not a number)

Jan2014

Computer Architecture

198

Dải giá trị biểu diễn

- 2^{-127} đến 2^{+127}
- 10^{-38} đến 10^{+38}

Jan2014

Computer Architecture

199

Dạng 64-bit

- S là bit dấu
- e (11 bit): mã *excess-1023* của phần mũ E $\rightarrow E = e - 1023$
- m (52 bit): phần lẻ của phần định trị M
- Giá trị số thực:

$$X = (-1)^S \cdot 1.m \cdot 2^{e-1023}$$
- Dải giá trị biểu diễn: 10^{-308} đến 10^{+308}

Jan2014

Computer Architecture

200

Dạng 128-bit

- S là bit dấu
- e (15 bit): mã **excess-16383** của phần mũ E $\rightarrow E = e - 16383$
- m (112 bit): phần lẻ của phần định trị M
- Giá trị số thực:

$$X = (-1)^S \cdot 1.m \cdot 2^{e-16383}$$
- Dải giá trị biểu diễn: 10^{-4932} đến 10^{+4932}

Jan2014

Computer Architecture

201

3. Thực hiện phép toán số dấu phẩy động

- $X_1 = M_1 \cdot R^{E_1}$
- $X_2 = M_2 \cdot R^{E_2}$
- Ta có
 - $X_1 \cdot X_2 = (M_1 \cdot M_2) \cdot R^{E_1+E_2}$
 - $X_1 / X_2 = (M_1 / M_2) \cdot R^{E_1-E_2}$
 - $X_1 \pm X_2 = (M_1 \cdot R^{E_1-E_2} \pm M_2) \cdot R^{E_2}$, với $E_2 \geq E_1$

Jan2014

Computer Architecture

202

Các khả năng tràn số

- Tràn trên số mũ (Exponent Overflow): mũ dương vượt ra khỏi giá trị cực đại của số mũ dương có thể. ($\rightarrow \infty$)
- Tràn dưới số mũ (Exponent Underflow): mũ âm vượt ra khỏi giá trị cực đại của số mũ âm có thể ($\rightarrow 0$).
- Tràn trên phần định trị (Mantissa Overflow): cộng hai phần định trị có cùng dấu, kết quả bị nhó ra ngoài bit cao nhất.
- Tràn dưới phần định trị (Mantissa Underflow): Khi hiệu chỉnh phần định trị, các số bị mất ở bên phải phần định trị.

Jan2014

Computer Architecture

203

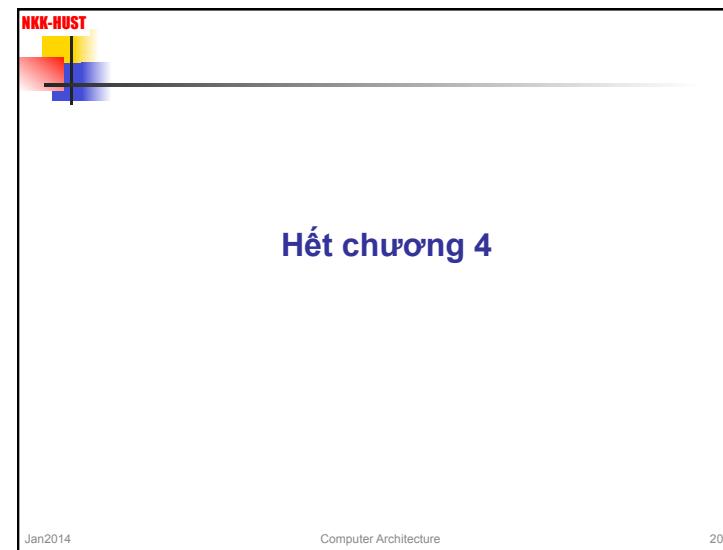
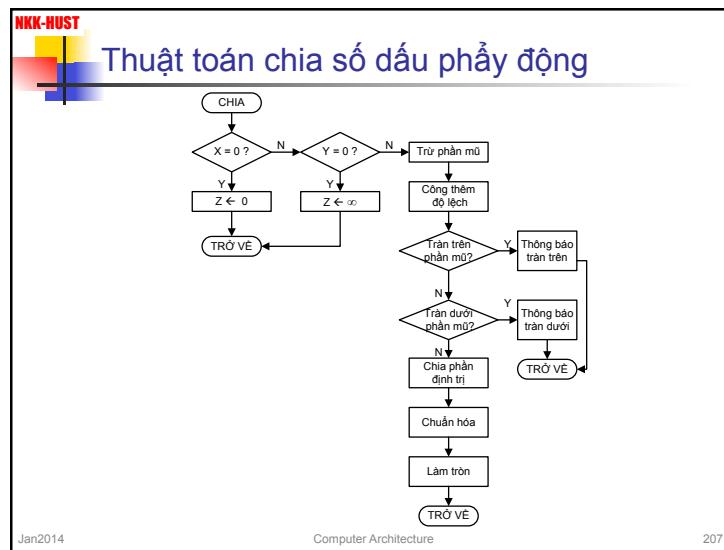
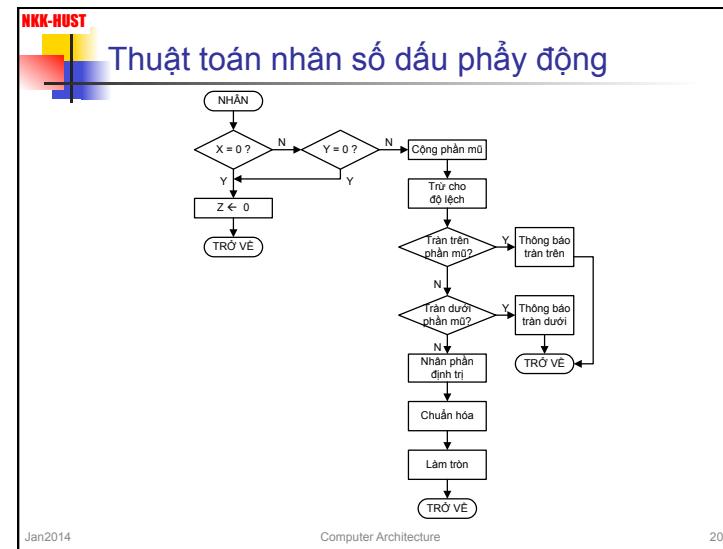
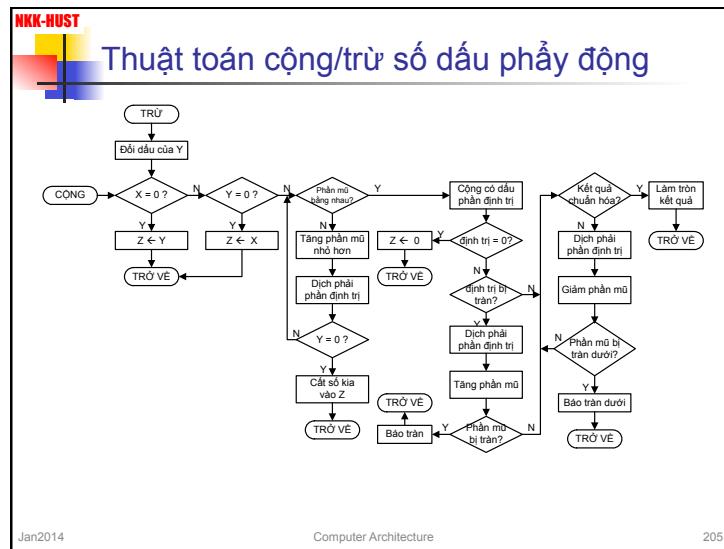
Phép cộng và phép trừ

- Kiểm tra các số hạng có bằng 0 hay không
- Hiệu chỉnh phần định trị
- Cộng hoặc trừ phần định trị
- Chuẩn hoá kết quả

Jan2014

Computer Architecture

204



NKK-HUST

Kiến trúc máy tính

Chương 5

KIẾN TRÚC TẬP LỆNH

(Instruction Set Architecture)

Nguyễn Kim Khánh
Trường Đại học Bách khoa Hà Nội

Jan2014 Computer Architecture 209

NKK-HUST

Nội dung học phần

- Chương 1. Giới thiệu chung
- Chương 2. Cơ bản về logic số
- Chương 3. Hệ thống máy tính
- Chương 4. Số học máy tính
- Chương 5. Kiến trúc tập lệnh**
- Chương 6. Bộ xử lý
- Chương 7. Bộ nhớ máy tính
- Chương 8. Hệ thống vào-ra
- Chương 9. Các kiến trúc song song

Jan2014 Computer Architecture 210

NKK-HUST

Nội dung của chương 5

- 5.1. Giới thiệu chung về kiến trúc tập lệnh
- 5.2. Kiến trúc tập lệnh MIPS
- 5.3. Kiến trúc tập lệnh Intel x86 *

Jan2014 Computer Architecture 211

NKK-HUST

5.1. Giới thiệu chung về kiến trúc tập lệnh

1. Mô hình lập trình của máy tính

PC: Program Counter
IR: Instruction Register

Jan2014 Computer Architecture 212

Tập thanh ghi

- Chứa các thông tin (dữ liệu, địa chỉ, trạng thái) cho hoạt động điều khiển và xử lý dữ liệu của CPU ở thời điểm hiện tại
- Được coi là mức đầu tiên của hệ thống nhớ
- Số lượng thanh ghi nhiều → tăng hiệu năng của CPU
- Có hai loại thanh ghi:
 - Các thanh ghi lập trình được
 - Các thanh ghi không lập trình được

Jan2014

Computer Architecture

213

Một số thanh ghi điển hình

- Bộ đếm chương trình PC (Program Counter)
- Con trỏ dữ liệu DP (Data Pointer)
- Con trỏ ngăn xếp SP (Stack Pointer)
- Thanh ghi cơ sở và Thanh ghi chỉ số (Base Register & Index Register)
- Các thanh ghi dữ liệu
- Thanh ghi trạng thái

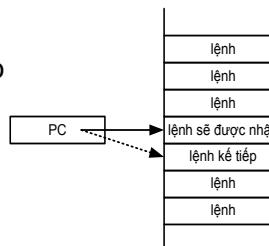
Jan2014

Computer Architecture

214

Bộ đếm chương trình PC

- Còn được gọi là con trỏ lệnh IP (Instruction Pointer)
- Giữ địa chỉ của lệnh tiếp theo sẽ được nhận vào.
- Sau khi một lệnh được nhận vào, nội dung PC **tự động tăng** để trỏ sang lệnh kế tiếp.
- PC tăng bao nhiêu?



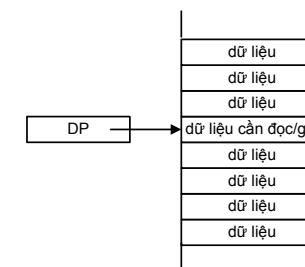
Jan2014

Computer Architecture

215

Thanh ghi con trỏ dữ liệu

- Chứa địa chỉ của ngăn nhớ dữ liệu mà CPU muốn truy nhập



Jan2014

Computer Architecture

216

Ngăn xếp (Stack)

- Ngăn xếp là vùng nhớ có cấu trúc LIFO (Last In - First Out → vào sau – ra trước)
- Ngăn xếp thường dùng để phục vụ cho chương trình con
- Đây ngăn xếp là một ngăn nhớ xác định
- Đỉnh ngăn xếp là thông tin nằm ở vị trí trên cùng trong ngăn xếp
- Đỉnh ngăn xếp có thể bị thay đổi

Jan2014 Computer Architecture 217

Con trỏ ngăn xếp SP (Stack Pointer)

- Chứa địa chỉ của ngăn nhớ đỉnh ngăn xếp
- Khi cất một thông tin vào ngăn xếp:
 - Nội dung của SP giảm
 - Thông tin được cất vào ngăn nhớ được trả bởi SP
- Khi lấy một thông tin ra khỏi ngăn xếp:
 - Thông tin được đọc từ ngăn nhớ được trả bởi SP
 - Nội dung của SP tăng
- Khi ngăn xếp rỗng, SP trở vào đáy

Jan2014 Computer Architecture 218

Thanh ghi cơ sở và thanh ghi chỉ số

- Để truy nhập một ngăn nhớ có thể sử dụng hai tham số:
 - Địa chỉ cơ sở (base address)
 - Phản dịch chuyển địa chỉ (offset)
 - Địa chỉ của ngăn nhớ cần truy nhập = địa chỉ cơ sở + offset
- Có thể sử dụng các thanh ghi để quản lý các tham số này:
 - Thanh ghi cơ sở: chứa địa chỉ cơ sở
 - Thanh ghi chỉ số: chứa phản dịch chuyển địa chỉ

Jan2014 Computer Architecture 219

Các thanh ghi dữ liệu

- Chứa các dữ liệu tạm thời hoặc các kết quả trung gian
- Cần có nhiều thanh ghi dữ liệu
- Các thanh ghi số nguyên: 8, 16, 32, 64 bit
- Các thanh ghi số dấu phẩy động

Jan2014 Computer Architecture 220

Thanh ghi trạng thái (Status Register)

- Được sử dụng trên một số kiến trúc cụ thể
- Còn gọi là thanh ghi cờ (Flag Register)
- Chứa các thông tin trạng thái của CPU
 - Các cờ phép toán: báo hiệu trạng thái của kết quả phép toán
 - Các cờ điều khiển: biểu thị trạng thái điều khiển của CPU

Jan2014

Computer Architecture

221

2. Thứ tự lưu trữ các byte trong bộ nhớ chính

- Bộ nhớ chính thường đánh địa chỉ theo byte
- Hai cách lưu trữ thông tin nhiều byte:
 - **Đầu nhỏ (Little-endian):** Byte có ý nghĩa thấp được lưu trữ ở ngăn nhớ có địa chỉ nhỏ, byte có ý nghĩa cao được lưu trữ ở ngăn nhớ có địa chỉ lớn.
 - **Đầu to (Big-endian):** Byte có ý nghĩa cao được lưu trữ ở ngăn nhớ có địa chỉ nhỏ, byte có ý nghĩa thấp được lưu trữ ở ngăn nhớ có địa chỉ lớn.

Jan2014

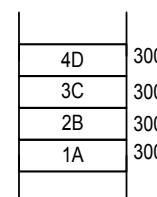
Computer Architecture

222

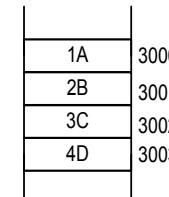
Ví dụ lưu trữ dữ liệu 32-bit

0001 1010 0010 1011 0011 1100 0100 1101

1A	2B	3C	4D
----	----	----	----



little-endian



big-endian

Jan2014

Computer Architecture

223

Lưu trữ của các bộ xử lý điển hình

- Intel x86: little-endian
- Motorola 680x0, MIPS, SunSPARC: big-endian
- Power PC, Itanium: bi-endian

Jan2014

Computer Architecture

224

NKK-HUST

3. Giới thiệu chung về tập lệnh

- Mỗi bộ xử lý có một tập lệnh xác định
- Tập lệnh thường có hàng chục đến hàng trăm lệnh
- Mỗi lệnh là một chuỗi số nhị phân mà bộ xử lý hiểu được để thực hiện một thao tác xác định.
- Các lệnh được mô tả bằng các ký hiệu gọi nhớ dạng text → chính là các lệnh của hợp ngữ (assembly language)

Jan2014 Computer Architecture 225

NKK-HUST

Các thành phần của lệnh máy

Mã thao tác	Địa chỉ của các toán hạng
-------------	---------------------------

- Mã thao tác (operation code → opcode): mã hóa cho thao tác mà bộ xử lý phải thực hiện
- Địa chỉ toán hạng: chỉ ra nơi chứa các toán hạng mà thao tác sẽ tác động
 - Toán hạng nguồn (source operand): dữ liệu vào của thao tác
 - Toán hạng đích (destination operand): dữ liệu ra của thao tác

Jan2014 Computer Architecture 226

NKK-HUST

Các kiểu thao tác thông dụng của tập lệnh

- Các lệnh chuyển dữ liệu
- Các lệnh xử lý số học
- Các lệnh xử lý logic
- Các lệnh chuyển điều khiển (rẽ nhánh, nhảy)

Jan2014 Computer Architecture 227

NKK-HUST

Định địa chỉ toán hạng

- Toán hạng của lệnh có thể là:
 - Một giá trị cụ thể nằm ngay trong lệnh
 - Nội dung của thanh ghi
 - Nội dung của ngăn nhớ hoặc cổng vào-ra
- Phương pháp định địa chỉ (addressing modes) là cách thức địa chỉ hóa trong trường địa chỉ của lệnh để xác định nơi chứa toán hạng

Jan2014 Computer Architecture 228

NKK-HUST

Các phương pháp định địa chỉ thông dụng

- Định địa chỉ tức thì
- Định địa chỉ thanh ghi
- Định địa chỉ trực tiếp
- Định địa chỉ gián tiếp qua thanh ghi
- Định địa chỉ dịch chuyển

Jan2014 Computer Architecture 229

NKK-HUST

Định địa chỉ tức thì

Mã thao tác		Toán hạng
-------------	--	-----------

- Toán hạng là hằng số nằm ngay trong lệnh
- Chỉ có thể là toán hạng nguồn
- Ví dụ:
ADD R1, 5 # R1 \leftarrow R1+5
- Không tham chiếu bộ nhớ
- Truy nhập toán hạng rất nhanh
- Dải giá trị của toán hạng bị hạn chế

Jan2014 Computer Architecture 230

NKK-HUST

Định địa chỉ thanh ghi

Mã thao tác		Tên thanh ghi
-------------	--	---------------

- Toán hạng nằm trong thanh ghi có tên được chỉ ra trong lệnh
- Ví dụ:
ADD R1, R2 # R1 \leftarrow R1+R2
- Số lượng thanh ghi ít \rightarrow Trường địa chỉ toán hạng chỉ cần ít bit
- Không tham chiếu bộ nhớ
- Truy nhập toán hạng nhanh
- Tăng số lượng thanh ghi \rightarrow hiệu quả hơn

Tập thanh ghi

Jan2014 Computer Architecture 231

NKK-HUST

Định địa chỉ trực tiếp

Mã thao tác		Địa chỉ
-------------	--	---------

- Toán hạng là ngăn nhớ có địa chỉ được cho trực tiếp trong lệnh
- Ví dụ:
ADD R1, A # R1 \leftarrow R1 + (A)
- Cộng nội dung thanh ghi R1 với nội dung của ngăn nhớ có địa chỉ là A
- Tìm toán hạng trong bộ nhớ ở địa chỉ A
- CPU tham chiếu bộ nhớ một lần để truy nhập dữ liệu

Bộ nhớ

Jan2014 Computer Architecture 232

Định địa chỉ gián tiếp qua thanh ghi

- Toán hạng nằm ở ngăn nhớ có địa chỉ trong thanh ghi
- Trường địa chỉ toán hạng cho biết tên thanh ghi đó
- Thanh ghi có thể là ngầm định
- Thanh ghi này được gọi là thanh ghi con trỏ
- Vùng nhớ có thể được tham chiếu là lớn (2^n), (với n là độ dài của thanh ghi)

Jan2014 Computer Architecture 233

Định địa chỉ dịch chuyển

- Để xác định toán hạng, Trường địa chỉ chứa hai thành phần:
 - Tên thanh ghi
 - Hàng số (offset)
- Địa chỉ của toán hạng = nội dung thanh ghi + hàng số
- Thanh ghi có thể được ngầm định

Jan2014 Computer Architecture 234

Số lượng địa chỉ toán hạng trong lệnh (1)

- Ba địa chỉ toán hạng:
 - 2 toán hạng nguồn, 1 toán hạng đích
 - Phù hợp với dạng: $c = a + b$
 - add r1, r2, r3 # $r1 \leftarrow r2 + r3$
 - Từ lệnh dài vì phải mã hoá địa chỉ cho cả ba toán hạng
 - Được sử dụng trên các bộ xử lý tiên tiến

Jan2014 Computer Architecture 235

Số lượng địa chỉ toán hạng trong lệnh (2)

- Hai địa chỉ toán hạng:
 - Một toán hạng vừa là toán hạng nguồn vừa là toán hạng đích; toán hạng còn lại là toán hạng nguồn
 - $a = a + b$
 - add r1, r2 # $r1 \leftarrow r1 + r2$
 - Giá trị cũ của 1 toán hạng nguồn bị mất vì phải chứa kết quả
 - Rút gọn độ dài từ lệnh
 - Phổ biến

Jan2014 Computer Architecture 236

NKK-HUST

Số lượng địa chỉ toán hạng trong lệnh (3)

- Một địa chỉ toán hạng:
 - Một toán hạng được chỉ ra trong lệnh
 - Một toán hạng là ngầm định → thường là thanh ghi (thanh chứa –accumulator)
 - add r1 # Acc ← Acc + r1
 - Được sử dụng trên các máy ở các thế hệ trước

Jan2014 Computer Architecture 237

NKK-HUST

Số lượng địa chỉ toán hạng trong lệnh (4)

- 0 địa chỉ toán hạng:
 - Các toán hạng đều được ngầm định
 - Sử dụng Stack
 - Ví dụ:
 - push a
 - push b
 - add
 - pop c
 - có nghĩa là: $c = a+b$
 - không thông dụng

Jan2014 Computer Architecture 238

NKK-HUST

4. CISC và RISC

- CISC: Complex Instruction Set Computer:
 - Máy tính với tập lệnh phức tạp
 - Các bộ xử lý truyền thống: Intel x86, Motorola 680x0
- RISC: Reduced Instruction Set Computer:
 - Máy tính với tập lệnh thu gọn
 - SunSPARC, Power PC, MIPS, ARM ...
 - RISC đối nghịch với CISC
 - Kiến trúc tập lệnh tiên tiến

Jan2014 Computer Architecture 239

NKK-HUST

Các đặc trưng của RISC

- Số lượng lệnh ít
- Hầu hết các lệnh truy nhập toán hạng ở các thanh ghi
- Truy nhập bộ nhớ bằng các lệnh LOAD/STORE
- Thời gian thực hiện lệnh là một chu kỳ máy
- Các lệnh có độ dài cố định (32 bit)
- Số lượng dạng lệnh ít (≤ 4)
- CPU có tập thanh ghi lớn
- Có ít phương pháp định địa chỉ toán hạng (≤ 4)
- Hỗ trợ các thao tác của ngôn ngữ bậc cao

Jan2014 Computer Architecture 240



5.2. Kiến trúc tập lệnh MIPS

Jan2014

Computer Architecture

241



1. Giới thiệu chung

- MIPS- Microprocessor without Interlocked Pipeline Stages
- Được phát triển ở đại học Stanford, sau đó được thương mại hóa bởi Công ty MIPS Technologies
- Năm 2012: MIPS Technologies được bán cho Imagination Technologies (imgtech.com)
- Kiến trúc RISC
- Chiếm thị phần lớn trong các sản phẩm nhúng
- Diễn hình cho nhiều kiến trúc tập lệnh hiện đại

Jan2014

Computer Architecture

242



2. Phép toán số học và các toán hạng

- Cộng và trừ: 3 toán hạng
 - Hai toán hạng nguồn và một toán hạng đích

add a, b, c # a ← b + c
- Tất cả các lệnh số học có dạng trên
- Toán hạng có thể là:
 - Nội dung thanh ghi
 - Nội dung ngắn nhớ
 - Hằng số

Jan2014

Computer Architecture

243



Toán hạng thanh ghi

- Các lệnh số học sử dụng toán hạng thanh ghi
- MIPS có tập 32 thanh ghi 32-bit
 - Được sử dụng thường xuyên
 - Được đánh số từ 0 đến 31 (dùng 5 bit)
 - Dữ liệu 32-bit được gọi là “word”
- Chương trình dịch Assembler đặt tên:
 - \$t0, \$t1, ..., \$t9 chứa các giá trị tạm thời
 - \$s0, \$s1, ..., \$s7 cất các biến

Jan2014

Computer Architecture

244

Tập thanh ghi của MIPS

Tên thanh ghi	Số hiệu thanh ghi	Công dụng
\$zero	0	the constant value 0
\$at	1	assembler temporary
\$v0-\$v1	2-3	procedure return values
\$a0-\$a3	4-7	procedure arguments
\$t0-\$t7	8-15	temporaries
\$s0-\$s7	16-23	saved variables
\$t8-\$t9	24-25	more temporaries
\$k0-\$k1	26-27	OS temporaries
\$gp	28	global pointer
\$sp	29	stack pointer
\$fp	30	frame pointer
\$ra	31	procedure return address

Jan2014

Computer Architecture

245

Ví dụ toán hạng thanh ghi

- Mã C:
 $a = b + c;$
 $d = a - e;$
- Mã hợp ngữ:
 $\text{add } a, b, c$
 $\text{sub } d, a, e$
- Mã C:
 $f = (g + h) - (i + j);$
 ■ f, g, h, i, j nằm ở \$s0, \$s1, \$s2, \$s3, \$s4
- Được dịch thành mã MIPS:
 $\text{add } \$t0, \$s1, \$s2 \quad \# \text{$t0} \leftarrow \$s1+\$s2$
 $\text{add } \$t1, \$s3, \$s4 \quad \# \text{$t1} \leftarrow \$s3+\$s4$
 $\text{sub } \$s0, \$t0, \$t1 \quad \# \text{$s0} \leftarrow \$t0-\$t1$

Jan2014

Computer Architecture

246

Toán hạng ở bộ nhớ

- Bộ nhớ chính được sử dụng lưu trữ các dữ liệu
 - Biến vô hướng, mảng, bản ghi, dữ liệu động
- Muốn thực hiện phép toán số học, cần phải:
 - Nạp (Load) các giá trị từ bộ nhớ vào các thanh ghi
 - Thực hiện phép toán trên các thanh ghi
 - Lưu (store) kết quả từ thanh ghi ra bộ nhớ
- Bộ nhớ được đánh địa chỉ theo byte
 - Mỗi địa chỉ xác định vị trí của một byte nhớ
- Mỗi Word có độ dài 32 bit \rightarrow địa chỉ của các Word là bội của 4
- MIPS lưu trữ theo kiểu Đầu to (Big Endian)

Jan2014

Computer Architecture

247

Ví dụ toán hạng bộ nhớ

- Mã C:
 $g = h + A[8];$
 ■ g ở \$s1, h ở \$s2, địa chỉ cơ sở của mảng A ở \$s3

Địa chỉ cơ sở	A[0]
	A[1]
	A[2]
	A[3]
	A[4]
	A[5]
	A[6]
	A[7]
	A[8]
	A[9]
	A[10]
	A[11]
	A[12]

Jan2014

Computer Architecture

248

NKK-HUST

Ví dụ toán hạng bộ nhớ

- Mã C:

$$g = h + A[8];$$
 - g ở $\$s1$, h ở $\$s2$, địa chỉ cơ sở của mảng A ở $\$s3$
- Mã MIPS:
 - Chỉ số 8 yêu cầu phần dịch chuyển (offset) là 32 (chỉ số từ 0) do 4 bytes/ word

```
lw $t0, 32($s3) # load word A[8]
add $s1, $s2, $t0 # g = h+A[8]
```

Jan2014 Computer Architecture 249

NKK-HUST

Ví dụ toán hạng bộ nhớ (tiếp)

- Mã C:

$$A[12] = h + A[8];$$
 - h ở $\$s2$, địa chỉ cơ sở của mảng A nằm ở $\$s3$

Jan2014 Computer Architecture 250

NKK-HUST

Ví dụ toán hạng bộ nhớ (tiếp)

- Mã C:

$$A[12] = h + A[8];$$
 - h ở $\$s2$, địa chỉ cơ sở của mảng A nằm ở $\$s3$
- Mã MIPS:


```
lw $t0, 32($s3) # load word from A[8]
add $t0, $s2, $t0 # t0 = h + A[8]
sw $t0, 48($s3) # store word to A[12]
```

Jan2014 Computer Architecture 251

NKK-HUST

Thanh ghi với Bộ nhớ

- Truy nhập thanh ghi nhanh hơn bộ nhớ
- Thao tác dữ liệu trên bộ nhớ yêu cầu nạp (load) và lưu (store).
 - Cần thực hiện nhiều lệnh hơn
- Chương trình dịch sử dụng các thanh ghi cho các biến nhiều nhất có thể
 - Chỉ sử dụng bộ nhớ cho các biến ít được sử dụng
 - Cần tối ưu hóa sử dụng thanh ghi

Jan2014 Computer Architecture 252

Toán hạng tức thì (immediate)

- Dữ liệu hằng số được xác định ngay trong lệnh

$$\text{addi } \$s3, \$s3, 4 \quad \# \quad \$s3 \leftarrow \$s3+4$$
- Không có lệnh trừ (subi) với giá trị tức thì
 - Sử dụng hằng số âm để thực hiện phép trừ

$$\text{addi } \$s2, \$s1, -1 \quad \# \quad \$s2 \leftarrow \$s1-1$$

Jan2014

Computer Architecture

253

Hằng số Zero

- Thanh ghi 0 của MIPS (\$zero hay \$0) luôn chứa hằng số 0
 - Không thể thay đổi giá trị
- Hữu ích cho một số thao tác thông dụng
 - Chẳng hạn, chuyển dữ liệu giữa các thanh ghi

$$\text{add } \$t2, \$s1, \$zero \quad \# \quad \$t2 \leftarrow \$s1$$

Jan2014

Computer Architecture

254

3. Mã máy

- Các lệnh được mã hóa dưới dạng nhị phân được gọi là mã máy
- Các lệnh của MIPS:
 - Được mã hóa bằng các từ lệnh 32-bit
 - Có ít dạng lệnh
- Số hiệu thanh ghi
 - \$t0 – \$t7 là các thanh ghi 8 – 15
 - \$t8 – \$t9 là các thanh ghi 24 – 25
 - \$s0 – \$s7 là các thanh ghi 16 – 23

Jan2014

Computer Architecture

255

Các dạng lệnh của MIPS

R	opcode	rs	rt	rd	shamt	funct	
	31	26 25	21 20	16 15	11 10	6 5	0
I	opcode	rs	rt			immediate	
	31	26 25	21 20	16 15			0
J	opcode				address		
	31	26 25					0

Jan2014

Computer Architecture

256

Lệnh dạng R (Register)

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Các trường của lệnh

- op: operation code (opcode): mã thao tác
- rs: số hiệu thanh ghi nguồn thứ nhất
- rt: số hiệu thanh ghi nguồn thứ hai
- rd: số hiệu thanh ghi đích
- shamt (shift amount): số bit được dịch
- funct: function code (extends opcode): mã hàm (mã thao tác mở rộng)

Jan2014

Computer Architecture

257

Ví dụ dạng lệnh R

Hợp ngữ

op	rs	rt	rd	shamt	funct
add \$s0, \$s1, \$s2	0	17	18	16	0
sub \$t0, \$t3, \$t5	0	11	13	8	0

Giá trị các trường

6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
--------	--------	--------	--------	--------	--------

Mã máy

op	rs	rt	rd	shamt	funct
000000	10001	10010	10000	00000	100000
000000	01011	01101	01000	00000	100010

Jan2014

Computer Architecture

258

Lệnh dạng I (Immediate)

op	rs	rt	constant or address
6 bits	5 bits	5 bits	16 bits

Dùng cho các lệnh số học với toán hạng tức thì và các lệnh load/store (nạp/lưu)

- rt: số hiệu thanh ghi đích hoặc thanh ghi nguồn
- Hằng số: từ -2^{15} đến $+2^{15} - 1$
- Địa chỉ: offset cộng với địa chỉ cơ sở nằm ở rs
- addi rt, rs, imm $\#(rt) \leftarrow (rs) + imm$
- lw rt, imm(rs) $\#(rt) \leftarrow mem[(rs) + imm]$
- sw rt, imm(rs) $\#(rt) \rightarrow mem[(rs) + imm]$

Jan2014

Computer Architecture

259

Ví dụ lệnh dạng I

Hợp ngữ

op	rs	rt	imm
addi \$s0, \$s1, 5	8	17	16
addi \$t0, \$s3, -12	8	19	8
lw \$t2, 32(\$0)	35	0	10
sw \$s1, 4(\$t1)	43	9	17

Giá trị các trường

6 bits	5 bits	5 bits	16 bits
--------	--------	--------	---------

Mã máy

op	rs	rt	imm
001000	10001	10000	0000 0000 0000 0101
001000	10011	01000	1111 1111 1111 0100
100011	00000	01010	0000 0000 0010 0000
101011	01001	10001	0000 0000 0000 0100

Jan2014

Computer Architecture

260

Lệnh lui

```
lui $s0, 61 # Giá trị tức thì 61 được nạp vào
# nửa cao của $s0 với 16 bit thấp
# được thiết lập về 0
```

Content of \$s0 after the instruction is executed

Jan2014 Computer Architecture 261

Lệnh kiểu J

- *Jump-type*
- Toán hạng 26-bit địa chỉ (addr)
- Được sử dụng cho các lệnh jump (j)

J-Type

op	addr
6 bits	26 bits

Jan2014 Computer Architecture 262

Các lệnh logic

- Các lệnh logic để thao tác trên các bit

Operation	C	Java	MIPS
Shift left	<<	<<	sll
Shift right	>>	>>	srl
Bitwise AND	&	&	and, andi
Bitwise OR			or, ori
Bitwise NOT	~	~	nor

Jan2014 Computer Architecture 263

Ví dụ lệnh logic

Source Registers		Result					
\$s1	1111 1111 1111 1111 0000 0000 0000 0000						
\$s2	0100 0110 1010 0001 1111 0000 1011 0111						
Assembly Code		Result					
and \$s3, \$s1, \$s2	\$s3						
or \$s4, \$s1, \$s2	\$s4						
xor \$s5, \$s1, \$s2	\$s5						
nor \$s6, \$s1, \$s2	\$s6						

Jan2014 Computer Architecture 264

NKK-HUST

Ví dụ lệnh logic

Source Registers

\$s1	1111	1111	1111	1111	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	
\$s2	0100	0110	1010	0001	1111	0000	1011	0111									

Assembly Code

and \$s3, \$s1, \$s2	\$s3	0100	0110	1010	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	
or \$s4, \$s1, \$s2	\$s4	1111	1111	1111	1111	1111	0000	1011	0111								
xor \$s5, \$s1, \$s2	\$s5	1011	1001	0101	1110	1111	0000	1011	0111								
nor \$s6, \$s1, \$s2	\$s6	0000	0000	0000	0000	0000	1111	0100	1000								

Result

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Jan2014

Computer Architecture

265

NKK-HUST

Ví dụ lệnh logic

Source Values

\$s1	0000	0000	0000	0000	0000	0000	1111	1111									
imm	0000	0000	0000	0000	1111	1010	0011	0100									

Assembly Code

andi \$s2, \$s1, 0xFA34	\$s2																
ori \$s3, \$s1, 0xFA34	\$s3																
xori \$s4, \$s1, 0xFA34	\$s4																

Result

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Jan2014

Computer Architecture

266

NKK-HUST

Ví dụ lệnh logic

Source Values

\$s1	0000	0000	0000	0000	0000	0000	1111	1111									
imm	0000	0000	0000	0000	1111	1010	0011	0100									

Assembly Code

andi \$s2, \$s1, 0xFA34	\$s2	0000	0000	0000	0000	0000	0011	0100									
ori \$s3, \$s1, 0xFA34	\$s3	0000	0000	0000	0000	1111	1010	1111	1111								
xori \$s4, \$s1, 0xFA34	\$s4	0000	0000	0000	0000	1111	1010	1100	1011								

Result

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Jan2014

Computer Architecture

267

- NKK-HUST**
- ### Ý nghĩa của các phép toán logic
- Phép AND dùng để giữ nguyên một số bit trong word, xóa các bit khác về 0
 - Phép OR dùng để giữ nguyên một số bit trong word, thiết lập các bit còn lại lên 1
 - Phép XOR dùng để giữ nguyên một số bit trong word, đảo giá trị các bit còn lại
 - Phép NOT dùng để đảo các bit trong word
 - Đổi 0 thành 1, và đổi 1 thành 0
 - MIPS không có lệnh NOT, nhưng có lệnh NOR với 3 toán hạng
 - $a \text{ NOR } b = a \text{ OR } b$
- nor \$t0, \$t1, \$zero
- Jan2014
- Computer Architecture
- 268

NKK-HUST

Thao tác dịch bit

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

- shamt: dịch bao nhiêu vị trí
- Dịch trái logic (shift left logical)
 - Dịch trái và điền các bit 0 vào bên phải
 - sll với i bits là nhân với 2^i
- Dịch phải logic (shift right logical)
 - Dịch phải và điền các bit 0 vào bên trái
 - srl với i bits là chia cho 2^i (chỉ với số nguyên không dấu)

Jan2014

Computer Architecture

269

NKK-HUST

Lệnh dịch

- sll: shift left logical
 - sll \$t0, \$t1, 5 # \$t0 <= \$t1 << 5
- srl: shift right logical
 - srl \$t0, \$t1, 5 # \$t0 <= \$t1 >> 5
- sra: shift right arithmetic
 - sra \$t0, \$t1, 5 # \$t0 <= \$t1 >>> 5

Variable shift instructions:

- sllv: shift left logical variable
 - sllv \$t0, \$t1, \$t2 # \$t0 <= \$t1 << \$t2
- srlv: shift right logical variable
 - sriv \$t0, \$t1, \$t2 # \$t0 <= \$t1 >> \$t2
- srav: shift right arithmetic variable
 - srav \$t0, \$t1, \$t2 # \$t0 <= \$t1 >>> \$t2

Jan2014

Computer Architecture

270

NKK-HUST

Ví dụ các lệnh dịch

Assembly Code						Field Values						
op	rs	rt	rd	shamt	funct	op	rs	rt	rd	shamt	funct	
sll \$t0, \$s1, 2	0	0	17	8	2	0	0	0	17	8	2	0
srl \$s2, \$s1, 2	0	0	17	18	2	2	0	0	17	18	2	2
sra \$s3, \$s1, 2	0	0	17	19	2	3	0	0	17	19	2	3

6 bits 5 bits 5 bits 5 bits 5 bits 6 bits

Machine Code

op	rs	rt	rd	shamt	funct
000000	00000	10001	01000	00010	000000 (0x00114080)
000000	00000	10001	10010	00010	000000 (0x00119082)
000000	00000	10001	10011	00010	000001 (0x00119883)

6 bits 5 bits 5 bits 5 bits 5 bits 6 bits

Jan2014

Computer Architecture

271

NKK-HUST

Nạp hằng số vào thanh ghi

- Trường hợp hằng số 16-bit → sử dụng lệnh addi:
 - Ví dụ: nạp hằng số 0x4f3c vào thanh ghi \$s0:


```
addi $s0, $0, 0x4f3c
```
- Trong trường hợp hằng số 32-bit → sử dụng lệnh lui và lệnh ori:
 - **lui rt, constant_hi16bit**
 - Copy 16 bit cao của hằng số vào 16 bit trái của rt
 - Xóa 16 bits bên phải của rt về 0
 - **ori rt, rt, constant_low16bit**
 - Đưa 16 bit thấp của hằng số 32 bit vào thanh ghi rt

Jan2014

Computer Architecture

272



Ví dụ khởi tạo thanh ghi 32-bit

- Nạp vào các thanh ghi \$s0 và \$s1 các giá trị 32-bit sau:

$$\begin{array}{cccccccc} 0010 & 0001 & 0001 & 0000 & 0000 & 0000 & 0011 & 1101 \\ 1111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 & 1111 \end{array}$$
- $0010\ 0001\ 0001\ 0000\ 0000\ 0000\ 0011\ 1101 = 0x2110003d$
 lui \$s0,0x2110

0010 0001 0001 0000	0000 0000 0000 0000
---------------------	---------------------

 ori \$s0,\$s0,0x003d

0010 0001 0001 0000	0000 0000 0011 1101
---------------------	---------------------
- $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111 = 0xffffffff$
 Có thể làm tương tự như trên với giá trị này, tuy nhiên có thể thực hiện đơn giản hơn:
 nor \$s1,\$zero,\$zero

Jan2014

Computer Architecture

273



4. Tạo các cấu trúc điều khiển

- Câu lệnh If
- Câu lệnh If/else
- Câu lệnh lặp While
- Câu lệnh lặp For

Jan2014

Computer Architecture

274



Các lệnh rẽ nhánh và lệnh nhảy

- Rẽ nhánh đến lệnh được đánh nhãn nếu điều kiện là đúng, ngược lại, thực hiện tuần tự
- bltz rs,L1
 - branch on less than zero
 - nếu ($rs < 0$) rẽ nhánh đến lệnh ở nhãn L1;
- beq rs, rt, L1
 - branch on equal
 - nếu ($rs == rt$) rẽ nhánh đến lệnh ở nhãn L1;
- bne rs, rt, L1
 - branch on not equal
 - nếu ($rs != rt$) rẽ nhánh đến lệnh ở nhãn L1;
- j L1
 - nhảy (jump) không điều kiện đến lệnh ở nhãn L1

Jan2014

Computer Architecture

275



Dịch câu lệnh If

- Mã C:


```
if (i==j)
  f = g+h;
  f = f-i;
  f, g, h, i, j ở $s0, $s1, $s2, $s3, $s4
```

Jan2014

Computer Architecture

276

Dịch câu lệnh If

- Mã C:


```
if (i==j)
        f = g+h;
        f = f-i;
      ■ f, g, h, i, j ở $s0, $s1, $s2, $s3, $s4
```
- Mã MIPS:


```
# $s0 = f, $s1 = g, $s2 = h
# $s3 = i, $s4 = j
      bne $s3, $s4, L1    # Nếu i=j
      add $s0, $s1, $s2    # thì f=g+h
L1: sub $s0, $s0, $s3  # f=f-i
```

Jan2014 Computer Architecture 277

Dịch câu lệnh If/else

- Mã C:


```
if (i==j) f = g+h;
else f = g-h;
      ■ f, g, h, i, j ở $s0, $s1, $s2, $s3, $s4
```

Jan2014 Computer Architecture 278

Dịch câu lệnh If/else

- Mã C:


```
if (i==j) f = g+h;
else f = g-h;
      ■ f, g, h, i, j ở $s0, $s1, $s2, $s3, $s4
```
- Mã MIPS:


```
bne $s3, $s4, Else # Nếu i=j
      add $s0, $s1, $s2    # thì f=g+h
      j Exit               # thoát
Else: sub $s0, $s1, $s2  # nếu khác
      ■ f = g-h
Exit: ...
```

Assembler calculates addresses

Jan2014 Computer Architecture 279

Dịch câu lệnh vòng lặp While

- Mã C:


```
while (save[i] == k) i += 1;
      ■ i ở $s3, k ở $s5, địa chỉ của mảng save ở $s6
```

Jan2014 Computer Architecture 280

Dịch câu lệnh vòng lặp While

- Mã C:


```
while (save[i] == k) i += 1;
      ■ i ở $s3, k ở $s5, địa chỉ của mảng save ở $s6
```
- Mã MIPS được dịch:


```
Loop: sll $t1,$s3,2    #$t1=4*i
          add $t1,$t1,$s6   #$t1 trả về save[i]
          lw $t0,0($t1)     #$t0 ← save[i]
          bne $t0,$s5,Exit  #nếu save[i]=k
          addi $s3,$s3,1    #thì i = i+1
          j Loop            #quay lại
Exit: ...              #nếu save[i]<>k, thoát
```

Jan2014 Computer Architecture 281

Dịch câu lệnh vòng lặp For

- Mã C:


```
// add the numbers from 0 to 9
int sum = 0;
int i;
for (i=0; i!=10; i = i+1) {
    sum = sum + i;
}
```

Jan2014 Computer Architecture 282

Dịch câu lệnh vòng lặp For

- Mã C:

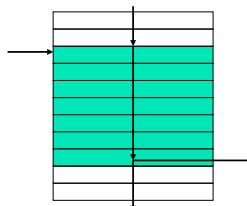

```
// add the numbers from 0 to 9
int sum = 0;
int i;
for (i=0; i!=10; i = i+1) {
    sum = sum + i;
}
```
- Mã MIPS được dịch:


```
# $s0 = i, $s1 = sum
      addi $s1, $0, 0      # sum = 0
      add $s0, $0, $0      # i = 0
      addi $t0, $0, 10     # $t0 = 10
for: beq $s0, $t0, done  # Nếu i = 10, thoát
      add $s1, $s1, $s0    # sum = sum + i
      addi $s0, $s0, 1      # i = i+1
      j for               # quay lại
done:
```

Jan2014 Computer Architecture 283

Khối lệnh cơ sở

- Khối lệnh cơ sở là dãy các lệnh với
 - Không có lệnh rẽ nhánh nhúng trong đó (ngoại trừ ở cuối)
 - Không có đích rẽ nhánh tới (ngoại trừ ở vị trí đầu tiên)
- Chương trình dịch xác định khối cơ sở để tối ưu hóa
- Các bộ xử lý tiên tiến có thể tăng tốc độ thực hiện khối cơ sở



Jan2014 Computer Architecture 284

Thêm các thao tác điều kiện

- Thiết lập kết quả = 1 nếu điều kiện là đúng, trái lại kết quả = 0
- **slt rd, rs, rt**
 - set on less than
 - if ($rs < rt$) rd = 1; else rd = 0;
- **slti rt, rs, constant**
 - if ($rs < \text{constant}$) rt = 1; else rt = 0;
- Sử dụng kết hợp với các lệnh beq, bne
 $\text{slt } \$t0, \$s1, \$s2 \quad \# \text{ if } (\$s1 < \$s2)$
 $\text{bne } \$t0, \$zero, L \quad \# \text{ branch to L}$

Jan2014

Computer Architecture

285

Ví dụ mã lệnh slt và slti

- **slt \$s1,\$s2,\$s3** # nếu $\$s2 < (\$s3)$, $\$s1 \leftarrow 1$
 # ngược lại $\$s1 \leftarrow 0$
- **slti \$s1,\$s2,61** # nếu $(\$s2) < 61$, $\$s1 \leftarrow 1$
 # ngược lại $\$s1 \leftarrow 0$

R	31	op	25	rs	20	rt	15	rd	10	sh	5	fm	0
ALU instruction	0 0 0 0 0 0	1 0 0 1 0	1 0 0 1 1	1 0 0 0 1	0 0 0 0 0	1 0 1 0 1	0						

I	31	op	25	rs	20	rt	15	operand / offset	0
slti = 10	0 0 1 0 1 0	1 0 0 1 0	1 0 0 0 1	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	1 1 1 0 1	0	

Jan2014

Computer Architecture

286

So sánh số có dấu và không dấu

- So sánh số có dấu: **slt, slti**
- So sánh số không dấu: **sltu, sltiu**
- Ví dụ
 - $\$s0 = 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111$
 - $\$s1 = 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001$
 - **slt \$t0, \$s0, \$s1 # signed**
 - $-1 < +1 \Rightarrow \$t0 = 1$
 - **sltu \$t0, \$s0, \$s1 # unsigned**
 - $+4,294,967,295 > +1 \Rightarrow \$t0 = 0$

Jan2014

Computer Architecture

287

5. Gọi thủ tục (Procedure Calling)

- Các bước yêu cầu:
 1. Đặt các tham số vào các thanh ghi
 2. Chuyển điều khiển đến thủ tục
 3. Thực hiện các thao tác của thủ tục
 4. Đặt kết quả vào thanh ghi cho chương trình đã gọi thủ tục
 5. Trở về vị trí đã gọi

Jan2014

Computer Architecture

288

Sử dụng các thanh ghi

- \$a0 – \$a3: các tham số (các thanh ghi 4 – 7)
- \$v0, \$v1: giá trị kết quả (các thanh ghi 2 và 3)
- \$t0 – \$t9: các giá trị tạm thời
 - Có thể được ghi lại bởi thủ tục được gọi
- \$s0 – \$s7: cất giữ các biến
 - Cần phải cắt/khôi phục bởi thủ tục được gọi
- \$gp: global pointer - con trỏ toàn cục cho dữ liệu tĩnh (thanh ghi 28)
- \$sp: stack pointer -con trỏ ngăn xếp (thanh ghi 29)
- \$fp: frame pointer – con trỏ khung (thanh ghi 30)
- \$ra: return address – địa chỉ trả về (thanh ghi 31)

Jan2014

Computer Architecture

289

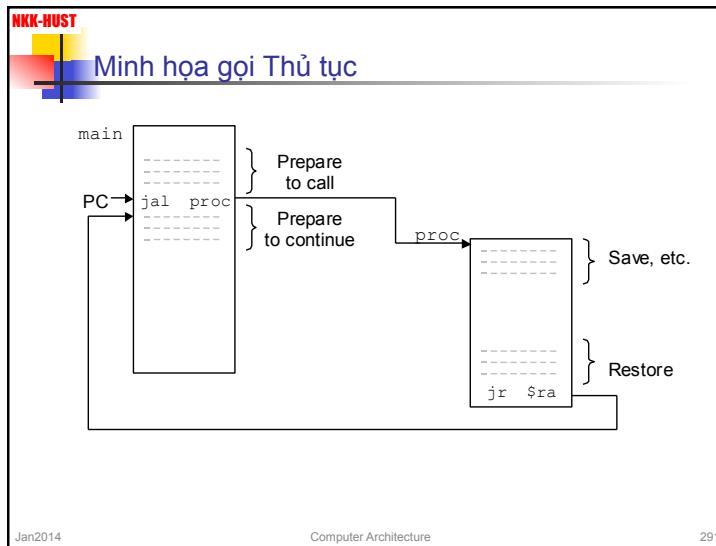
Các lệnh gọi thủ tục

- Gọi thủ tục: jump and link
jal ProcedureLabel
 - Địa chỉ của lệnh kế tiếp được cất ở \$ra
 - Nhảy đến nhãn đích
- Trở về từ thủ tục: jump register
jr \$ra
 - Copy \$ra vào bộ đếm chương trình PC

Jan2014

Computer Architecture

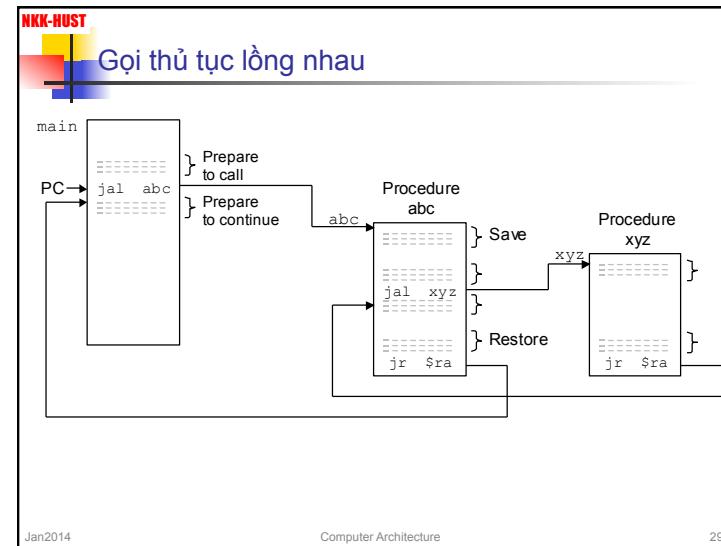
290



Jan2014

Computer Architecture

291



Jan2014

Computer Architecture

292

Ví dụ Thủ tục lá

- Thủ tục lá là thủ tục không có lời gọi thủ tục khác
- Mã C:

```
int leaf_example (int g, h, i, j)
{ int f;
  f = (g + h) - (i + j);
  return f;
}
  ■ Các tham số g, h, i, j ở $a0, $a1, $a2, $a3
  ■ f ở $s0 (do đó, cần cất $s0 ra ngăn xếp)
  ■ Kết quả ở $v0
```

Jan2014

Computer Architecture

293

Ví dụ Thủ tục lá

- Mã MIPS:

leaf_example:

addi \$sp, \$sp, -4	
sw \$s0, 0(\$sp)	Cất \$s0 ra stack
add \$t0, \$a0, \$a1	Thân thủ tục
add \$t1, \$a2, \$a3	Kết quả
sub \$s0, \$t0, \$t1	Khôi phục \$s0
add \$v0, \$s0, \$zero	Trở về
lw \$s0, 0(\$sp)	
addi \$sp, \$sp, 4	
jr \$ra	

Jan2014

Computer Architecture

294

Ví dụ Thủ tục cành

- Là thủ tục có gọi thủ tục khác
- C code:

```
int fact (int n)
{
  if (n < 1) return (1);
  else return n * fact(n - 1);
}
  ■ Tham số n ở $a0
  ■ Kết quả ở $v0
```

Jan2014

Computer Architecture

295

Ví dụ Thủ tục cành (tiếp)

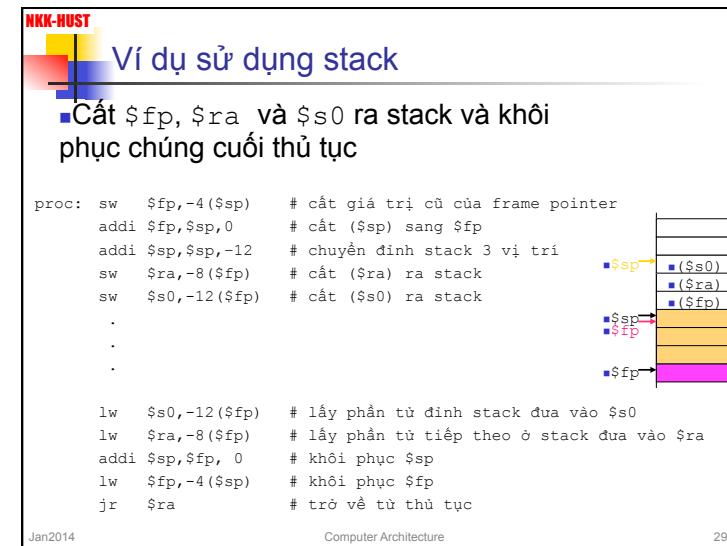
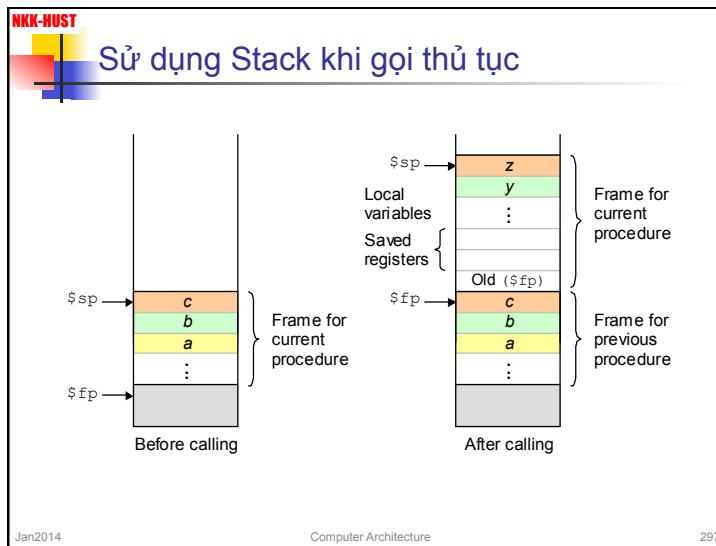
- Mã MIPS:

fact:	
addi \$sp, \$sp, -8	# dành stack cho 2 mục
sw \$ra, 4(\$sp)	# cất địa chỉ trả về
sw \$a0, 0(\$sp)	# cất tham số n
slti \$t0, \$a0, 1	# kiểm tra n < 1
beq \$t0, \$zero, L1	
addi \$v0, \$zero, 1	# nếu đúng, kết quả là 1
addi \$sp, \$sp, 8	# lấy 2 mục từ stack
jr \$ra	# và trả về
L1: addi \$a0, \$a0, -1	# nếu không, giảm n
jal fact	# gọi đệ quy
lw \$a0, 0(\$sp)	# khôi phục n ban đầu
lw \$ra, 4(\$sp)	# và địa chỉ trả về
addi \$sp, \$sp, 8	# lấy 2 mục từ stack
mul \$v0, \$a0, \$v0	# nhân để nhận kết quả
jr \$ra	# và trả về

Jan2014

Computer Architecture

296



- NKK-HUST**
- ### Dữ liệu ký tự
- Các tập ký tự được mã hóa theo byte
 - ASCII: 128 ký tự
 - 95 ký thị hiển thị, 33 mã điều khiển
 - Latin-1: 256 ký tự
 - ASCII và các ký tự mở rộng
 - Unicode: Tập ký tự 32-bit
 - Được sử dụng trong Java, C++, ...
 - Hầu hết các ký tự của các ngôn ngữ trên thế giới và các ký hiệu
- Jan2014 Computer Architecture 299

- NKK-HUST**
- ### Các thao tác với Byte/Halfword
- Có thể sử dụng các phép toán logic
 - Nạp/Lưu byte/halfword trong MIPS
 - lb rt, offset(rs) lh rt, offset(rs)
 - Mở rộng dấu thành 32 bits trong rt
 - lbu rt, offset(rs) lhu rt, offset(rs)
 - Mở rộng zero thành 32 bits trong rt
 - sb rt, offset(rs) sh rt, offset(rs)
 - Chỉ lưu byte/halfword bên phải
- Jan2014 Computer Architecture 300

Ví dụ copy String

- Mã C:

```
void strcpy (char x[], char y[])
{ int i;
  i = 0;
  while ((x[i]=y[i])!='\0')
    i += 1;
}
```

- Các địa chỉ của x, y ở \$a0, \$a1
- i ở \$s0

Jan2014

Computer Architecture

301

Ví dụ Copy String

- MIPS code:

strcpy:	
addi	\$sp, \$sp, -4 # adjust stack for 1 item
sw	\$s0, 0(\$sp) # save \$s0
add	\$s0, \$zero, \$zero # i = 0
L1: add	\$t1, \$s0, \$a1 # addr of y[i] in \$t1
lbu	\$t2, 0(\$t1) # \$t2 = y[i]
add	\$t3, \$s0, \$a0 # addr of x[i] in \$t3
sb	\$t2, 0(\$t3) # x[i] = y[i]
beq	\$t2, \$zero, L2 # exit loop if y[i] == 0
addi	\$s0, \$s0, 1 # i = i + 1
j	L1 # next iteration of loop
L2: lw	\$s0, 0(\$sp) # restore saved \$s0
addi	\$sp, \$sp, 4 # pop 1 item from stack
jr	\$ra # and return

Jan2014

Computer Architecture

302

Địa chỉ hóa cho các lệnh Branch

- Các lệnh Branch chỉ ra:
 - Mã thao tác, hai thanh ghi, offset
- Hầu hết các đích rẽ nhánh là rẽ nhánh gần
 - Rẽ xuôi hoặc rẽ ngược

op	rs	rt	constant
6 bits	5 bits	5 bits	16 bits

- Định địa chỉ tương đối với PC
 - PC-relative addressing
 - Địa chỉ đích = PC + hằng số × 4
 - Chú ý: trước đó PC đã được tăng lên

Jan2014

Computer Architecture

303

Ví dụ lệnh Branch

- bltz \$s1,L # rẽ nhánh khi (\$s1) < 0
- beq \$s1,\$s2,L # rẽ nhánh khi (\$s1) = (\$s2)
- bne \$s1,\$s2,L # rẽ nhánh khi (\$s1) ≠ (\$s2)

31	op	25	rs	20	rt	15	operand / offset	0
0	0 0 0 0 1	1 0 0 0 1	0 0 0 0 0	0 0 0 0 0 0 0 0 0 1 1 1 1 0 1	bltz = 1	Source	Zero	Relative branch distance in words

31	op	25	rs	20	rt	15	operand / offset	0	
0	0 0 0 1 0	x	1 0 0 0 1	1 0 0 1 0	0 0 0 0 0 0 0 0 0 1 1 1 1 0 1	beq = 4	Source 1	Source 2	Relative branch distance in words

Jan2014

Computer Architecture

304

NKK-HUST

Địa chỉ hóa cho lệnh Jump

- Đích của lệnh **Jump (j và jal)** có thể là bất kỳ chỗ nào trong chương trình
 - Cần mã hóa đầy đủ địa chỉ trong lệnh

op	address
6 bits	26 bits

- Định địa chỉ nhảy (giả) trực tiếp (Pseudo)Direct jump addressing
 - Địa chỉ đích = $PC_{31\ldots28} : (address \times 4)$

Jan2014 Computer Architecture 305

NKK-HUST

Ví dụ mã lệnh Jump

- j verify # nhảy đến vị trí có nhãn “verify”
- jr \$ra # nhảy đến vị trí có địa chỉ ở \$ra; # \$ra may hold a return address

From PC Effective target address (32 bits)

J 31 op 25 jump target address 0
j = 2

R 31 op 25 rs 20 rt 15 rd 10 sh 5 fn 0
ALU instruction Source register Unused Unused Unused jr = 8

\$ra là thanh ghi \$31 (return address)

Jan2014 Computer Architecture 306

NKK-HUST

Ví dụ mã hóa lệnh

```

Loop: sll $t1, $s3, 2    80000 0 0 19 9 2 0
      add $t1, $t1, $s6 80004 0 9 22 9 0 32
      lw $t0, 0($t1)   80008 35 9 8 0
      bne $t0, $s5, Exit 80012 5 8 21 2
      addi $s3, $s3, 1   80016 8 19 19 1
      j Loop            80020 2 20000
Exit: ...
  
```

A hex dump of the assembly code is shown below the assembly listing. The first row has columns 0, 0, 19, 9, 2, 0. The second row has columns 0, 9, 22, 9, 0, 32. The third row has columns 35, 9, 8, 0. The fourth row has columns 5, 8, 21, 2. The fifth row has columns 8, 19, 19, 1. The sixth row has columns 2, 20000.

Jan2014 Computer Architecture 307

NKK-HUST

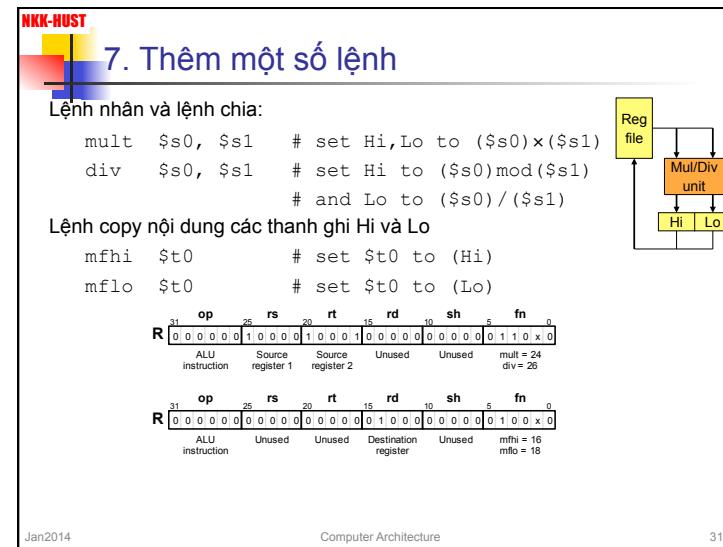
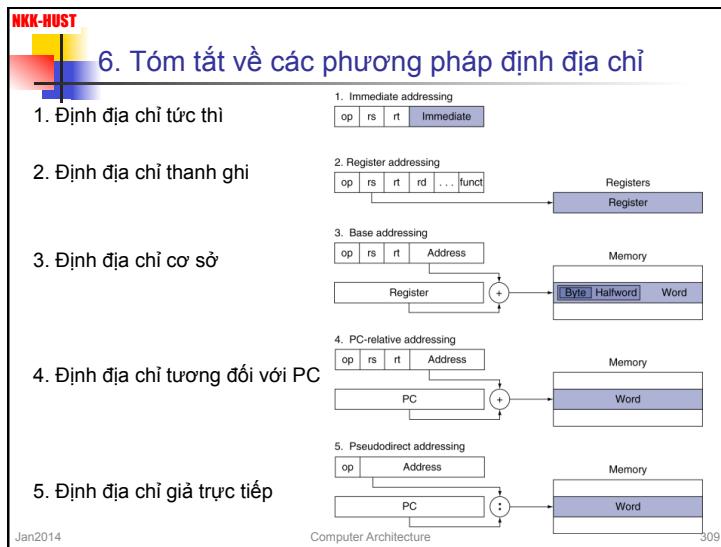
Rẽ nhánh xa

- Nếu đích rẽ nhánh là quá xa để mã hóa với offset 16-bit, assembler sẽ viết lại code
- Ví dụ

```

beq $s0,$s1, L1
      ↓
      bne $s0,$s1, L2
      j L1
L2: ...
  
```

Jan2014 Computer Architecture 308



NKK-HUST

Các lệnh số học số nguyên không dấu

```
addu $t0,$s0,$s1    # set $t0 to ($s0)+($s1)
subu $t0,$s0,$s1    # set $t0 to ($s0)-($s1)
multu $s0,$s1        # set Hi,Lo to ($s0)×($s1)
divu $s0,$s1        # set Hi to ($s0)mod($s1)
                  # and Lo to ($s0)/($s1)
addiu $t0,$s0,61     # set $t0 to ($s0)+61;
                  # the immediate operand is
                  # sign extended
```

Jan2014 Computer Architecture 311

NKK-HUST

Các lệnh với số dấu phẩy động

- Các thanh ghi số dấu phẩy động
 - 32 thanh ghi 32-bit (single-precision): \$f0, \$f1, ... \$f31
 - Cặp đôi để chứa dữ liệu dạng 64-bit (double-precision): \$f0/\$f1, \$f2/\$f3, ...
 - (Release 2 of MIPS ISA supports 32 × 64-bit FP reg's)
- Các lệnh số dấu phẩy động chỉ thực hiện trên các thanh ghi số dấu phẩy động
- FP load and store instructions
 - lwc1, ldc1, swc1, sdc1
 - e.g., ldc1 \$f8, 32(\$sp)

Jan2014 Computer Architecture 312

NKK-HUST

Các lệnh với số dấu phẩy động

- Single-precision arithmetic
 - add.s, sub.s, mul.s, div.s
 - e.g., add.s \$f0, \$f1, \$f6
- Double-precision arithmetic
 - add.d, sub.d, mul.d, div.d
 - e.g., mul.d \$f4, \$f4, \$f6
- Single- and double-precision comparison
 - c.xx.s, c.xx.d (xx is eq, lt, le, ...)
 - Sets or clears FP condition-code bit
 - e.g. c.lt.s \$f3, \$f4
- Branch on FP condition code true or false
 - bc1t, bc1f
 - e.g., bc1t TargetLabel

Jan2014 Computer Architecture 313

NKK-HUST

8. Thực hành lập trình hợp ngữ MIPS

- Phần mềm lập trình: MARS
- Lập trình các ví dụ
- Chạy các chương trình có sẵn và phân tích
- MIPS Reference Data

Jan2014 Computer Architecture 314

NKK-HUST

Dịch và chạy ứng dụng

```

graph TD
    A[High Level Code] --> B[Compiler]
    B --> C[Assembly Code]
    C --> D[Assembler]
    D --> E[Object File]
    E --> F[Linker]
    F --> G[Executable]
    F --> H[Object Files  
Library Files]
    H --> G
    G --> I[Loader]
    I --> J[Memory]
  
```

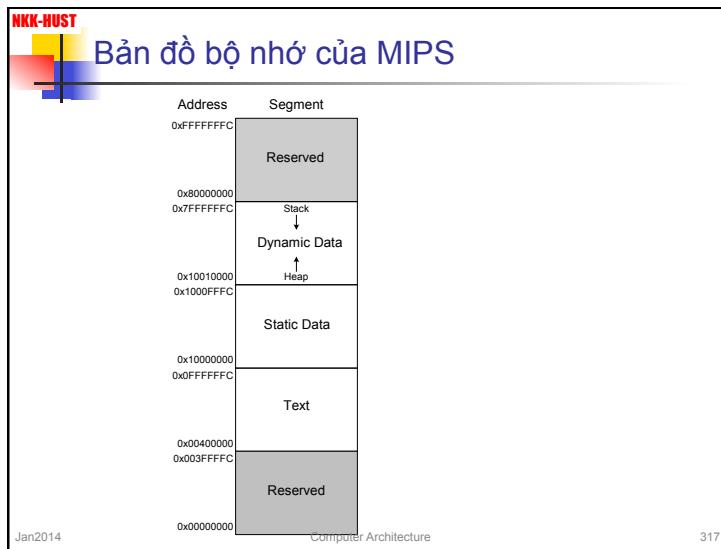
Jan2014 Computer Architecture 315

NKK-HUST

Chương trình trong bộ nhớ

- Các lệnh (instructions)
- Dữ liệu
 - Toàn cục/tĩnh: được cấp phát trước khi chương trình bắt đầu thực hiện
 - Động: được cấp phát trong khi chương trình thực hiện
- Bộ nhớ:
 - $2^{32} = 4$ gigabytes (4 GB)
 - Từ địa chỉ 0x00000000 đến 0xFFFFFFFF

Jan2014 Computer Architecture 316



NKK-HUST

Ví dụ: Mã C

```
int f, g, y; // global variables

int main(void)
{
    f = 2;
    g = 3;
    y = sum(f, g);
    return y;
}

int sum(int a, int b) {
    return (a + b);
}
```

Jan2014 Computer Architecture 318

NKK-HUST

Ví dụ chương trình hợp ngữ

```
.data
f:
g:
y:
.text
main:
    addi $sp, $sp, -4    # stack frame
    sw $ra, 0($sp)      # store $ra
    addi $a0, $0, 2      # $a0 = 2
    sw $a0, f            # f = 2
    addi $a1, $0, 3      # $a1 = 3
    sw $a1, g            # g = 3
    jal sum              # call sum
    sw $v0, y             # y = sum()
    lw $ra, 0($sp)      # restore $ra
    addi $sp, $sp, 4      # restore $sp
    jr $ra                # return to OS
sum:
    add $v0, $a0, $a1    # $v0 = a + b
    jr $ra                # return
```

Jan2014 Computer Architecture 319

NKK-HUST

Bảng ký hiệu

Ký hiệu	Địa chỉ
f	0x10000000
g	0x10000004
y	0x10000008
main	0x00400000
sum	0x0040002C

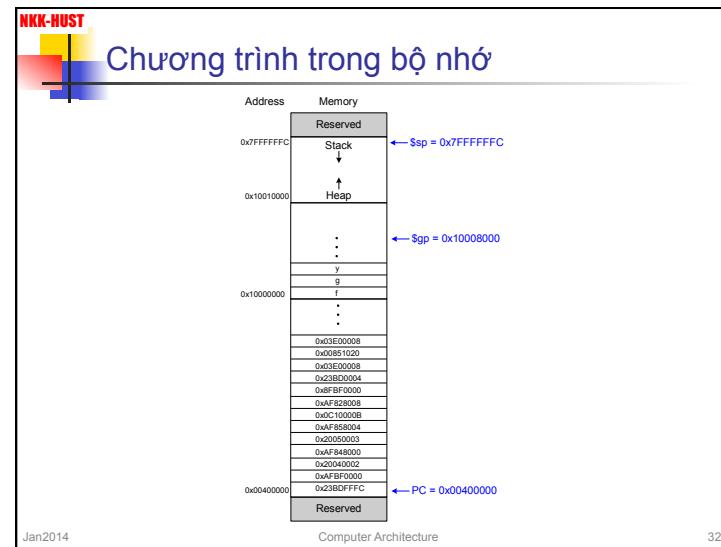
Jan2014 Computer Architecture 320

NKK-HUST

Chương trình thực thi

Executable file header	Text Size	Data Size
	0x34 (52 bytes)	0xC (12 bytes)
Text segment	Address	Instruction
	0x00400000	0x23BDFFFF
	0x00400004	sw \$ra, 0 (\$sp)
	0x00400008	addi \$a0, \$0, 2
	0x0040000C	sw \$a0, 0x8008 (\$gp)
	0x00400010	addi \$a1, \$0, 3
	0x00400014	sw \$a1, 0x8004 (\$gp)
	0x00400018	jal 0x0040002C
	0x0040001C	sw \$v0, 0x8008 (\$gp)
	0x00400020	lw \$ra, 0 (\$sp)
	0x00400024	addi \$sp, \$sp, -4
	0x00400028	jr \$ra
	0x0040002C	add \$v0, \$a0, \$a1
	0x00400030	jr \$ra
Data segment	Address	Data
	0x10000000	f
	0x10000004	g
	0x10000008	y

Jan2014 Computer Architecture 321



NKK-HUST

Ví dụ lệnh giả (Pseudoinstruction)

Pseudoinstruction	MIPS Instructions
li \$s0, 0x1234AA77	lui \$s0, 0x1234 ori \$s0, 0xAA77
mul \$s0, \$s1, \$s2	mult \$s1, \$s2 mflo \$s0
clear \$t0	add \$t0, \$0, \$0
move \$s1, \$s2	add \$s2, \$s1, \$0
nop	sll \$0, \$0, 0

Jan2014 Computer Architecture 323

- NKK-HUST**
- ### 5.3. Kiến trúc tập lệnh Intel x86(*)
- Sự tiến hóa của các bộ xử lý Intel
 - 8080 (1974): 8-bit microprocessor
 - Accumulator, plus 3 index-register pairs
 - 8086 (1978): 16-bit extension to 8080
 - Complex instruction set (CISC)
 - 8087 (1980): floating-point coprocessor
 - Adds FP instructions and register stack
 - 80286 (1982): 24-bit addresses, MMU
 - Segmented memory mapping and protection
 - 80386 (1985): 32-bit extension (now IA-32)
 - Additional addressing modes and operations
 - Paged memory mapping as well as segments
- Jan2014 Computer Architecture 324

NKK-HUST

Kiến trúc tập lệnh Intel x86

- i486 (1989): pipelined, on-chip caches and FPU
 - Compatible competitors: AMD, Cyrix, ...
- Pentium (1993): superscalar, 64-bit datapath
 - Later versions added MMX (Multi-Media eXtension) instructions
 - The infamous FDIV bug
- Pentium Pro (1995), Pentium II (1997)
 - New microarchitecture
- Pentium III (1999)
 - Added SSE (Streaming SIMD Extensions) and associated registers
- Pentium 4 (2001)
 - New microarchitecture
 - Added SSE2 instructions
- Intel Core (2006)
 - Added SSE4 instructions, virtual machine support

Jan2014 Computer Architecture 325

NKK-HUST

Các thanh ghi cơ bản của x86

Name	Use
EAX	GPR 0
ECX	GPR 1
EDX	GPR 2
EBX	GPR 3
ESP	GPR 4
EBP	GPR 5
ESI	GPR 6
EDI	GPR 7
CS	Code segment pointer
SS	Stack segment pointer (top of stack)
DS	Data segment pointer 0
ES	Data segment pointer 1
FS	Data segment pointer 2
GS	Data segment pointer 3
EIP	Instruction pointer (PC)
EFLAGS	Condition codes

Jan2014 Computer Architecture 326

NKK-HUST

Các phương pháp định địa chỉ cơ bản

- Hai toán hạng của lệnh

Source/dest operand	Second source operand
Register	Register
Register	Immediate
Register	Memory
Memory	Register
Memory	Immediate

- Các phương pháp định địa chỉ
 - Address in register
 - Address = $R_{base} + displacement$
 - Address = $R_{base} + 2^{scale} \times R_{index}$ ($scale = 0, 1, 2, \text{ or } 3$)
 - Address = $R_{base} + 2^{scale} \times R_{index} + displacement$

Jan2014 Computer Architecture 327

NKK-HUST

Mã hóa lệnh x86

- a. JE EIP + displacement

4	4	8
JE	Condition	Displacement
- b. CALL

8	32
CALL	Offset
- c. MOV EBX, [EDI + 45]

6	1	1	8	8	
MOV	d	w	r/m	Postbyte	Displacement
- d. PUSH ESI

5	3
PUSH	Reg
- e. ADD EAX, #6765

4	3	1	32
ADD	Reg	w	Immediate
- f. TEST EDX, #42

7	1	8	32
TEST	w	Postbyte	Immediate

Jan2014 Computer Architecture 328



Hết chương 5

Jan2014 Computer Architecture 329



Kiến trúc máy tính

Chương 6 BỘ XỬ LÝ (Processor)

Nguyễn Kim Khánh
Trường Đại học Bách khoa Hà Nội

Jan2014 Computer Architecture 330



Nội dung học phần

- Chương 1. Giới thiệu chung
- Chương 2. Cơ bản về logic số
- Chương 3. Hệ thống máy tính
- Chương 4. Số học máy tính
- Chương 5. Kiến trúc tập lệnh
- Chương 6. Bộ xử lý**
- Chương 7. Bộ nhớ máy tính
- Chương 8. Hệ thống vào-ra
- Chương 9. Các kiến trúc song song

Jan2014 Computer Architecture 331



Nội dung

- 6.1. Tổ chức của bộ xử lý
- 6.2. Thiết kế đơn vị điều khiển
- 6.3. Kỹ thuật đường ống lệnh
- 6.4. Ví dụ thiết kế bộ xử lý theo kiến trúc MIPS*

Jan2014 Computer Architecture 332

NKK-HUST

6.1. Tổ chức của CPU

1. Cấu trúc cơ bản của CPU

- Nhiệm vụ của CPU:
 - Nhận lệnh (Fetch Instruction): CPU đọc lệnh từ bộ nhớ.
 - Giải mã lệnh (Decode Instruction): xác định thao tác mà lệnh yêu cầu.
 - Nhận dữ liệu (Fetch Data): nhận dữ liệu từ bộ nhớ hoặc các cổng vào-ra.
 - Xử lý dữ liệu (Process Data): thực hiện phép toán số học hay phép toán logic với các dữ liệu.
 - Ghi dữ liệu (Write Data): ghi dữ liệu ra bộ nhớ hay cổng vào-ra.

Jan2014 Computer Architecture 333

NKK-HUST

Sơ đồ cấu trúc cơ bản của CPU

```

graph TD
    CU[Đơn vị điều khiển CU] --- IBU[Đơn vị nối ghép bus BIU]
    ALU[Đơn vị số học và logic ALU] --- IBU
    RF[Tập thanh ghi RF] --- IBU
    IBU -- bus điều khiển --> CU
    IBU -- bus dữ liệu --> ALU
    IBU -- bus địa chỉ --> RF
  
```

The diagram illustrates the basic structure of a CPU. It features four main functional units arranged vertically: the Control Unit (CU) in pink, the Arithmetic and Logic Unit (ALU) in light blue, the Register File (RF) in light green, and the Bus Interface Unit (BIU) in yellow. The CU, ALU, and RF are interconnected by a central internal bus (bus bên trong). The BIU is connected to the CU, ALU, and RF via three separate buses: the control bus (bus điều khiển), the data bus (bus dữ liệu), and the address bus (bus địa chỉ).

Jan2014 Computer Architecture 334

NKK-HUST

Các thành phần cơ bản của CPU

- Đơn vị điều khiển (Control Unit - CU)
- Đơn vị số học và logic (Arithmetic and Logic Unit - ALU)
- Tập thanh ghi (Register File - RF)
- Đơn vị nối ghép bus (Bus Interface Unit - BIU)
- Bus bên trong (Internal Bus)

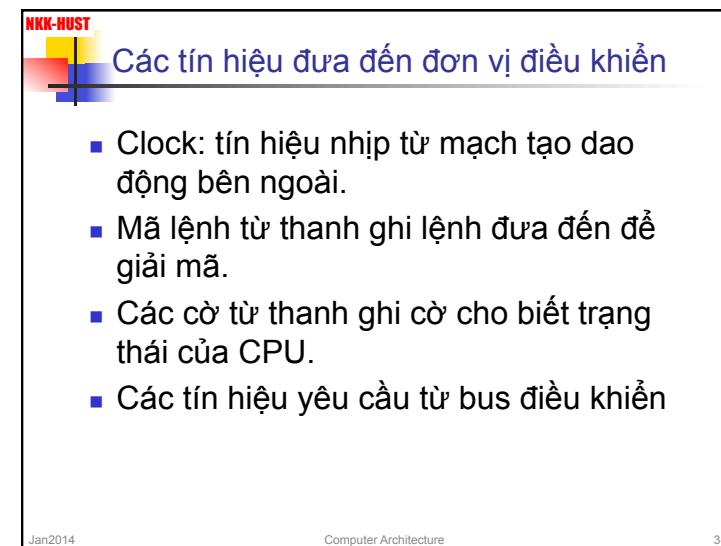
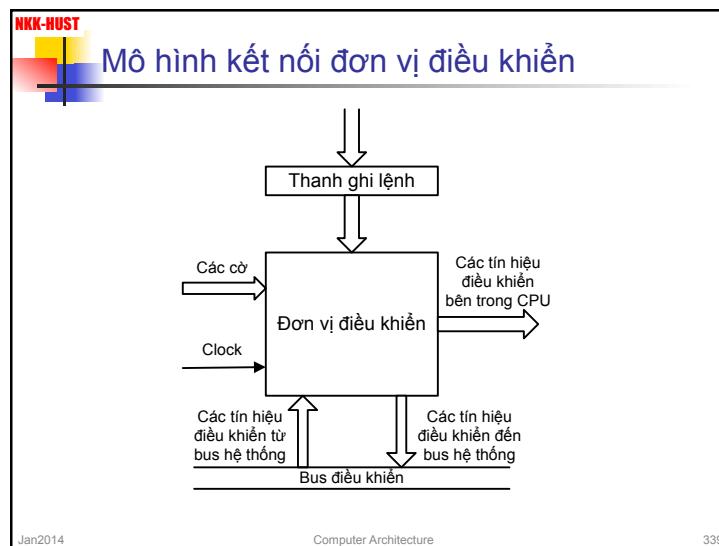
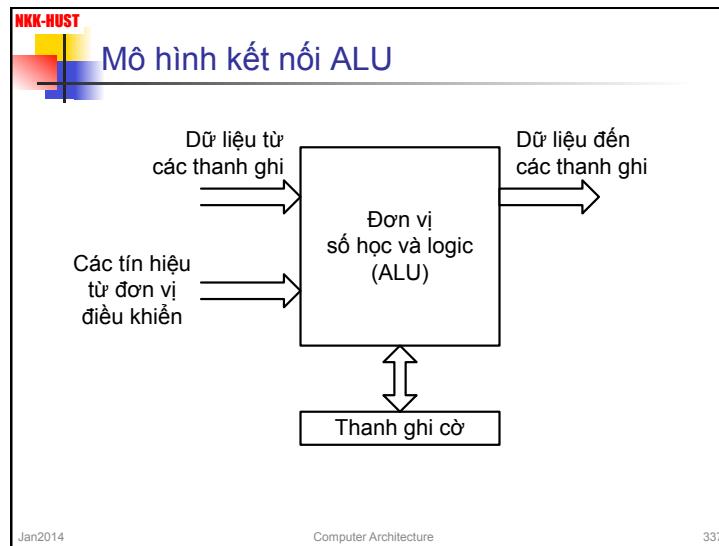
Jan2014 Computer Architecture 335

NKK-HUST

2. Đơn vị số học và logic

- **Chức năng:** Thực hiện các phép toán số học và phép toán logic:
 - Số học: cộng, trừ, nhân, chia, tăng, giảm, đảo dấu
 - Logic: AND, OR, XOR, NOT, phép dịch bit.

Jan2014 Computer Architecture 336



NKK-HUST

Các tín hiệu phát ra từ đơn vị điều khiển

- Các tín hiệu điều khiển bên trong CPU:
 - Điều khiển các thanh ghi
 - Điều khiển ALU
- Các tín hiệu điều khiển bên ngoài CPU:
 - Điều khiển bộ nhớ
 - Điều khiển các mô-đun vào-ra

Jan2014 Computer Architecture 341

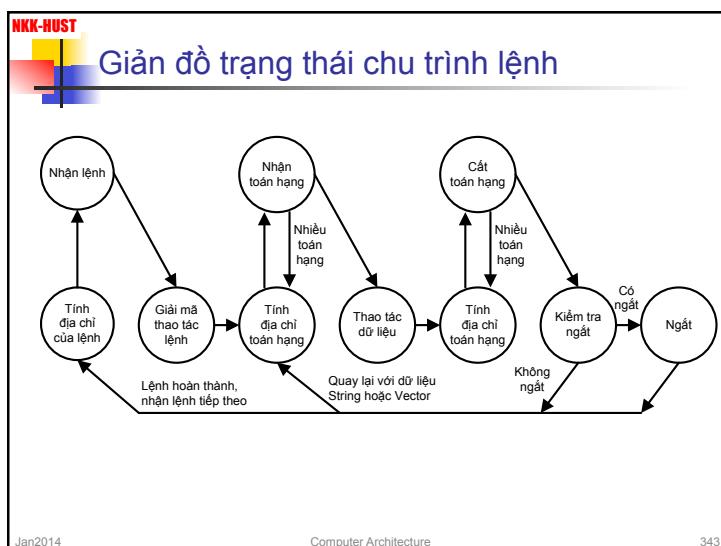
NKK-HUST

4. Hoạt động của chu trình lệnh

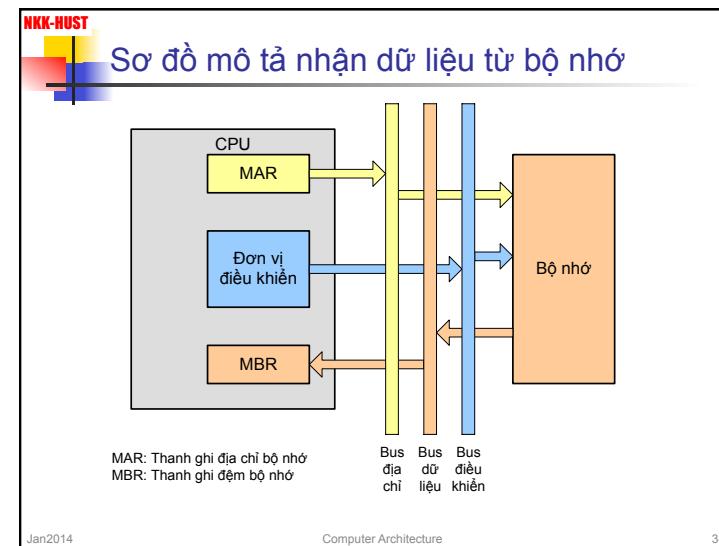
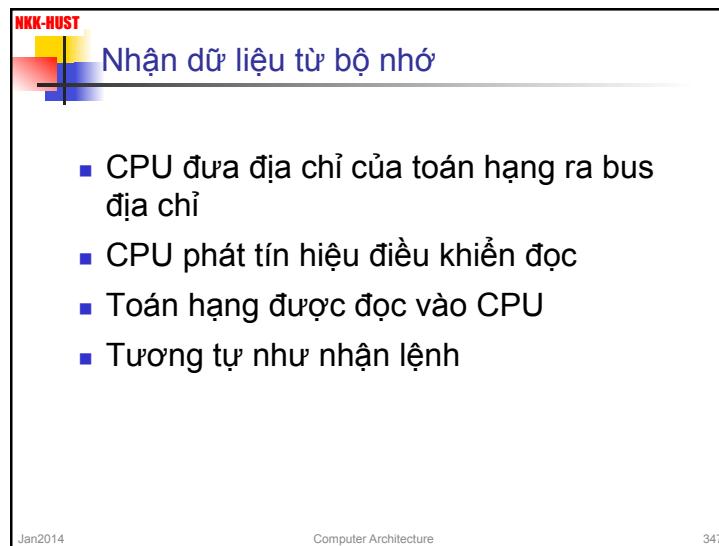
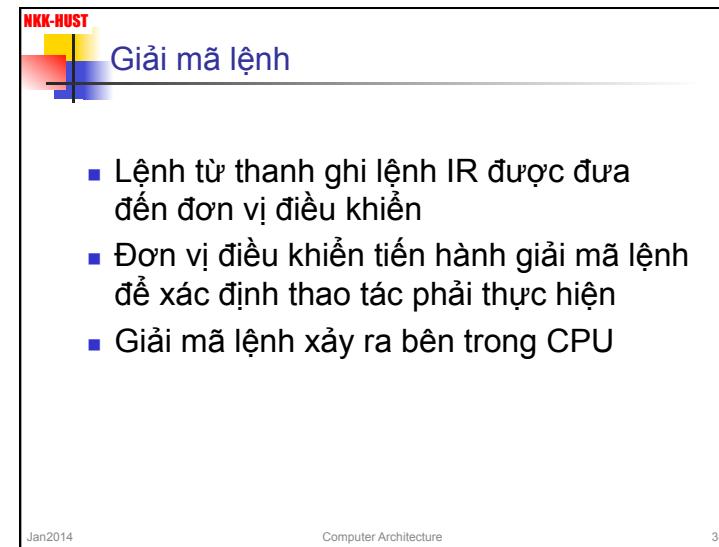
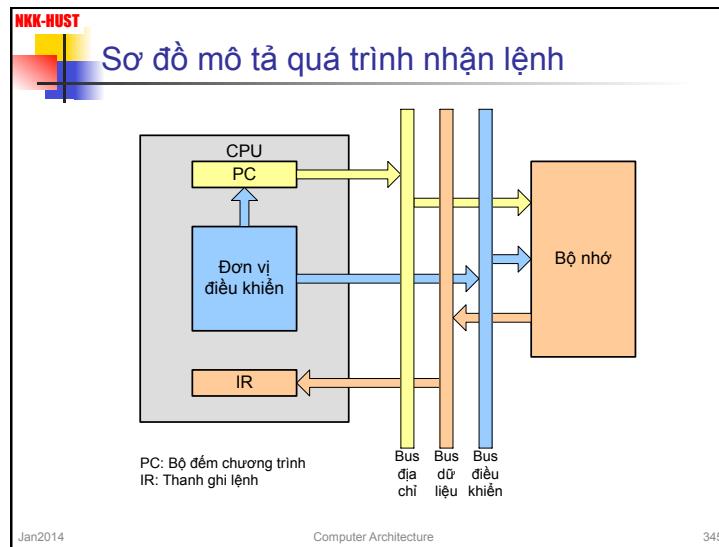
Chu trình lệnh

- Nhận lệnh
- Giải mã lệnh
- Nhận toán hạng
- Thực hiện lệnh
- Cắt toán hạng
- Ngắt

Jan2014 Computer Architecture 342



- NKK-HUST**
- ### Nhận lệnh
- CPU đưa địa chỉ của lệnh cần nhận từ bộ đếm chương trình PC ra bus địa chỉ
 - CPU phát tín hiệu điều khiển đọc bộ nhớ
 - Lệnh từ bộ nhớ được đặt lên bus dữ liệu và được CPU copy vào thanh ghi lệnh IR
 - CPU tăng nội dung PC để trỏ sang lệnh kế tiếp
- Jan2014 Computer Architecture 344



NKK-HUST

Thực hiện lệnh

- Có nhiều dạng tuỳ thuộc vào lệnh
- Có thể là:
 - Đọc/Ghi bộ nhớ
 - Vào/Ra
 - Chuyển giữa các thanh ghi
 - Thao tác số học/logic
 - Chuyển điều khiển (rẽ nhánh)
 - ...

Jan2014 Computer Architecture 349

NKK-HUST

Ghi toán hạng

- CPU đưa địa chỉ ra bus địa chỉ
- CPU đưa dữ liệu cần ghi ra bus dữ liệu
- CPU phát tín hiệu điều khiển ghi
- Dữ liệu trên bus dữ liệu được copy đến vị trí xác định

Jan2014 Computer Architecture 350

NKK-HUST

Sơ đồ mô tả quá trình ghi toán hạng

MAR: Thanh ghi địa chỉ bộ nhớ
MBR: Thanh ghi đệm bộ nhớ

Bus địa chỉ, Bus dữ liệu, Bus điều khiển

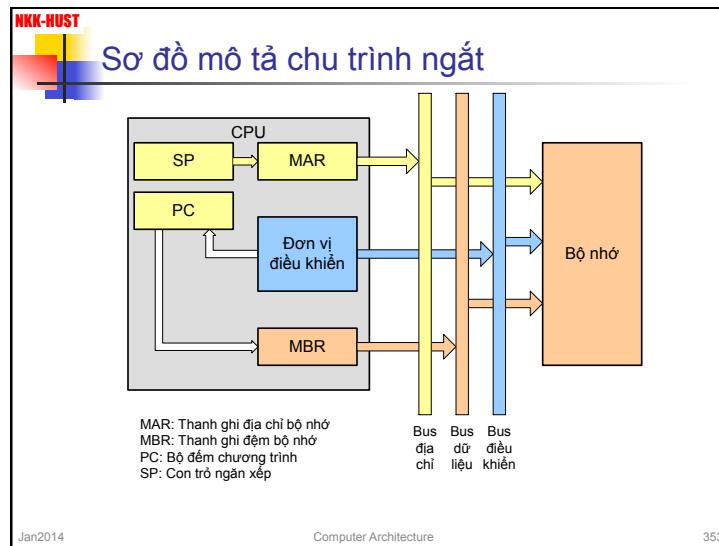
Jan2014 Computer Architecture 351

NKK-HUST

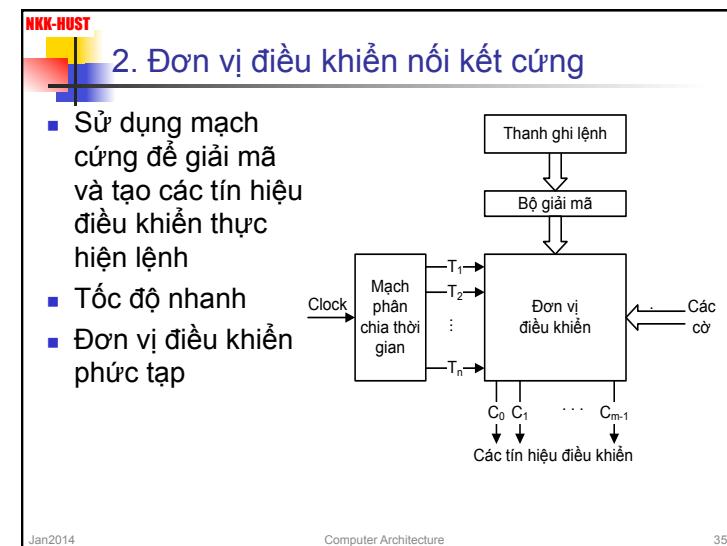
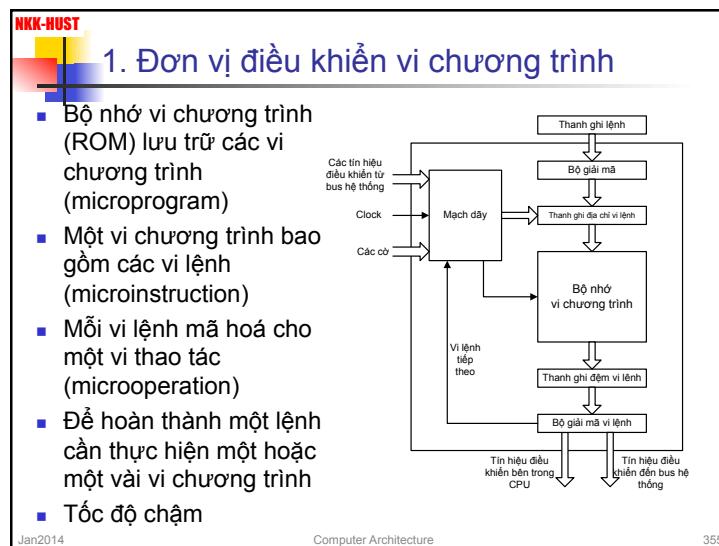
Ngắt

- Nội dung của bộ đệm chương trình PC (địa chỉ trả về sau khi ngắt) được đưa ra bus dữ liệu
- CPU đưa địa chỉ (thường được lấy từ con trỏ ngăn xếp SP) ra bus địa chỉ
- CPU phát tín hiệu điều khiển ghi bộ nhớ
- Địa chỉ trả về trên bus dữ liệu được ghi ra vị trí xác định (ở ngăn xếp)
- Địa chỉ lệnh đầu tiên của chương trình con điều khiển ngắt được nạp vào PC

Jan2014 Computer Architecture 352



- NKK-HUST**
- ### 6.2. Các phương pháp thiết kế đơn vị điều khiển
- Đơn vị điều khiển vi chương trình (Microprogrammed Control Unit)
 - Đơn vị điều khiển nối kết cứng (Hardwired Control Unit)
- Jan2014 Computer Architecture 354

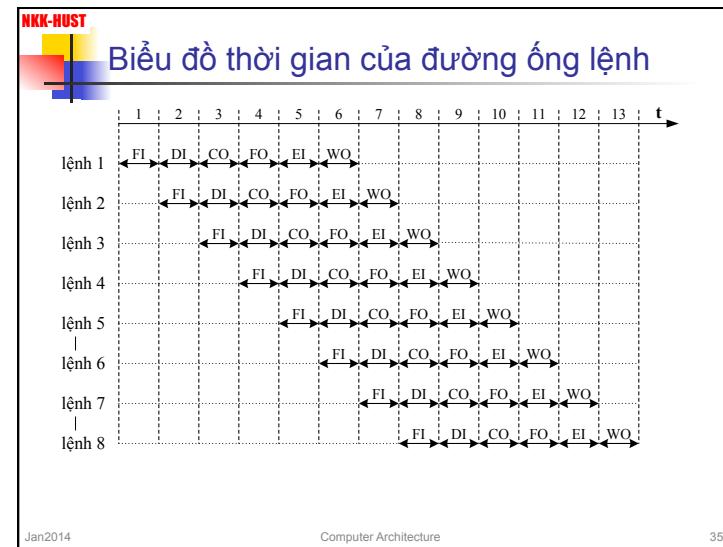


NKK-HUST

6.3. Kỹ thuật đường ống lệnh và song song mức lệnh

- Kỹ thuật đường ống lệnh (Instruction Pipelining): Chia chu trình lệnh thành các công đoạn và cho phép thực hiện gối lên nhau (như dây chuyền lắp ráp)
- Chẳng hạn có 6 công đoạn:
 - Nhận lệnh (Fetch Instruction - FI)
 - Giải mã lệnh (Decode Instruction - DI)
 - Tính địa chỉ toán hạng (Calculate Operand Address-CO)
 - Nhận toán hạng (Fetch Operands - FO)
 - Thực hiện lệnh (Execute Instruction - EI)
 - Ghi toán hạng (Write Operands - WO)

Jan2014 Computer Architecture 357



NKK-HUST

Các Hazard (trở ngại) của đường ống lệnh

- Hazard: Tình huống ngăn cản bắt đầu của lệnh tiếp theo ở chu kỳ tiếp theo.
 - Hazard cấu trúc: do tài nguyên được yêu cầu đang bận
 - Hazard dữ liệu: cần phải đợi để lệnh trước hoàn thành việc đọc/ghi dữ liệu
 - Hazard điều khiển: do rẽ nhánh gây ra

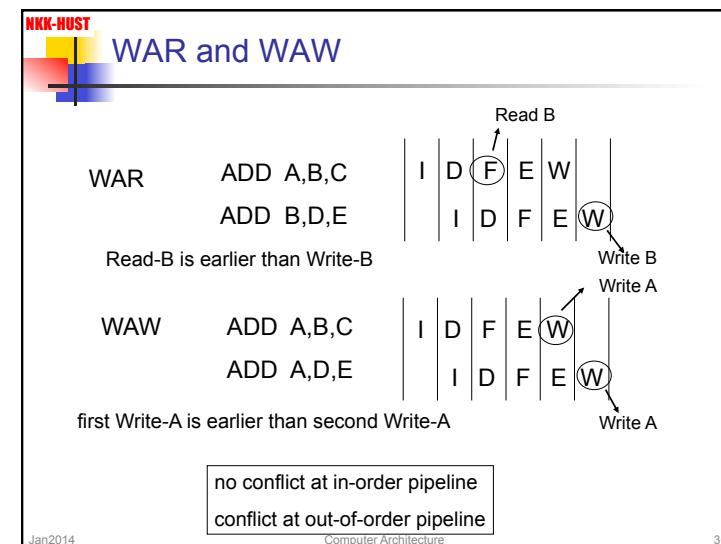
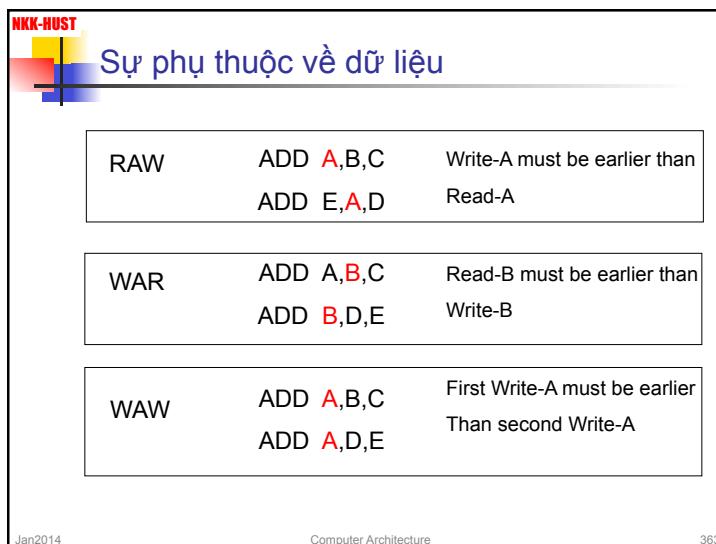
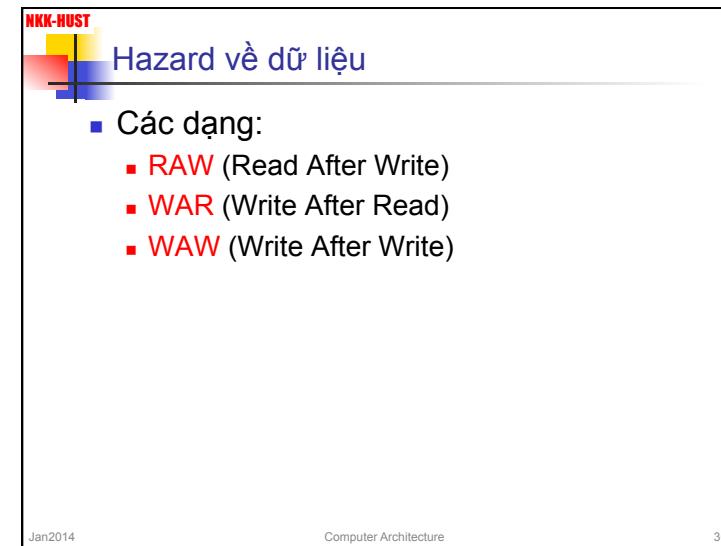
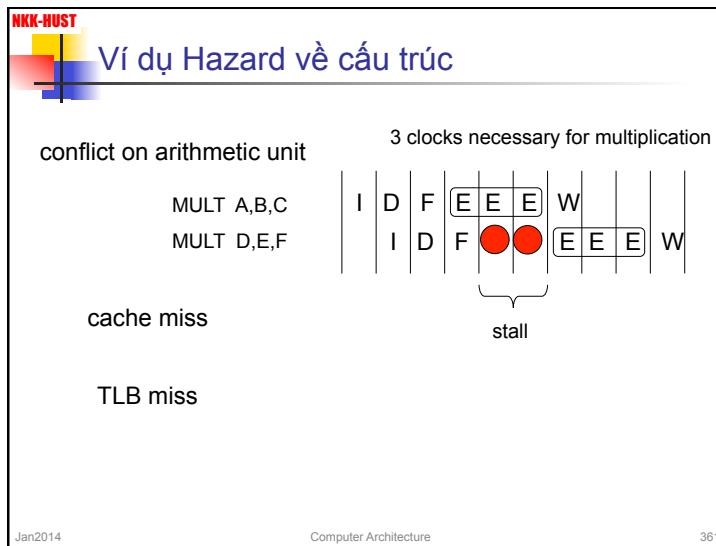
Jan2014 Computer Architecture 359

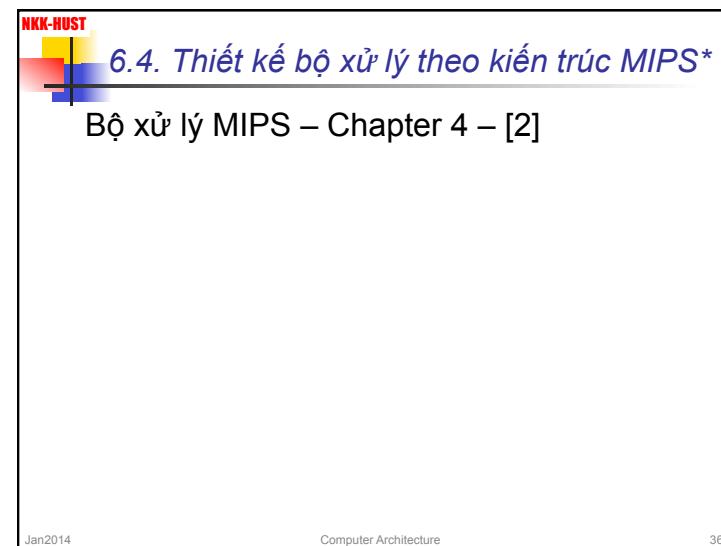
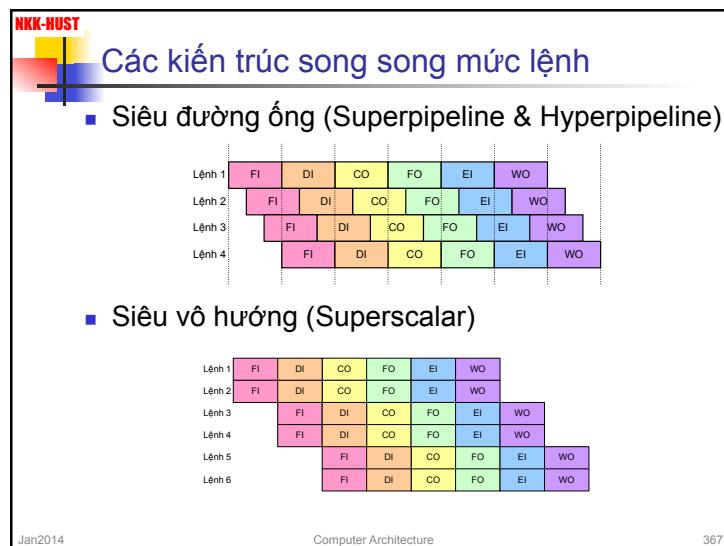
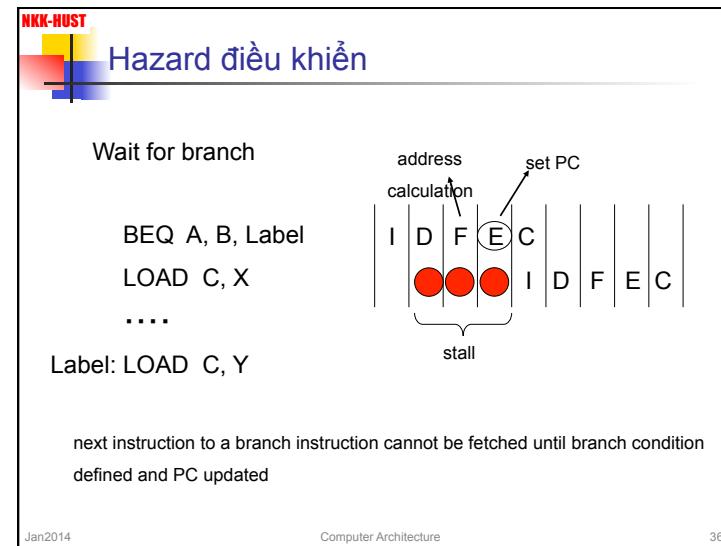
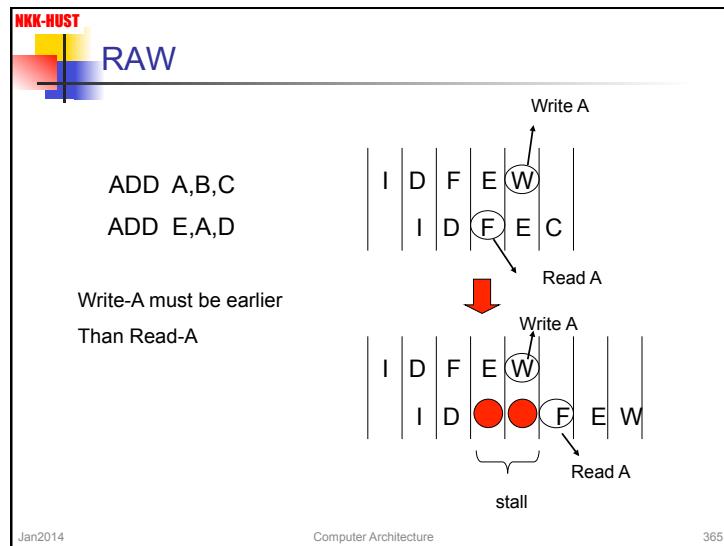
NKK-HUST

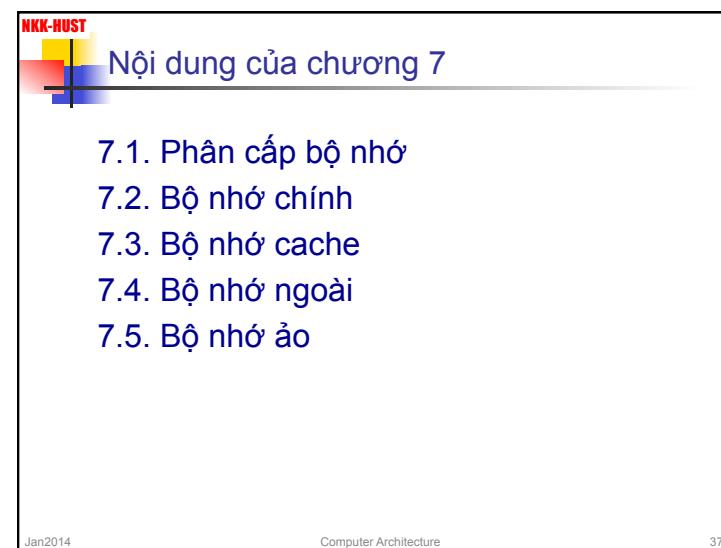
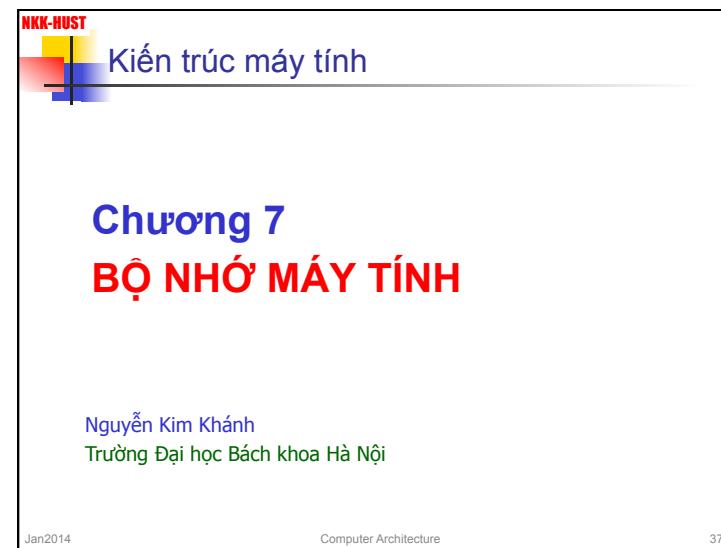
Hazard về cấu trúc

- Khắc phục:
 - nhân tài nguyên để tránh xung đột
 - Làm trễ
- Ví dụ:
 - Bus dữ liệu: truyền lệnh và dữ liệu
→ Bus lệnh riêng, bus dữ liệu riêng (cache lệnh và cache dữ liệu)

Jan2014 Computer Architecture 360







7.1. Phân cấp bộ nhớ máy tính

1. Các đặc trưng của bộ nhớ máy tính

- Vị trí
 - Bên trong CPU:
 - tập thanh ghi
 - Bộ nhớ trong:
 - bộ nhớ chính
 - bộ nhớ cache
 - Bộ nhớ ngoài: các thiết bị nhớ
- Dung lượng
 - Độ dài từ nhớ (tính bằng bit)
 - Số lượng từ nhớ

Jan2014

Computer Architecture

373

Các đặc trưng của hệ thống nhớ (tiếp)

- Đơn vị truyền
 - Từ nhớ
 - Khối nhớ
- Phương pháp truy nhập
 - Truy nhập tuần tự (băng từ)
 - Truy nhập trực tiếp (các loại đĩa)
 - Truy nhập ngẫu nhiên (bộ nhớ bán dẫn)
 - Truy nhập liên kết (cache)

Jan2014

Computer Architecture

374

Các đặc trưng của hệ thống nhớ (tiếp)

- Hiệu năng (performance)
 - Thời gian truy nhập
 - Chu kỳ nhớ
 - Tốc độ truyền
- Kiểu vật lý
 - Bộ nhớ bán dẫn
 - Bộ nhớ từ
 - Bộ nhớ quang

Jan2014

Computer Architecture

375

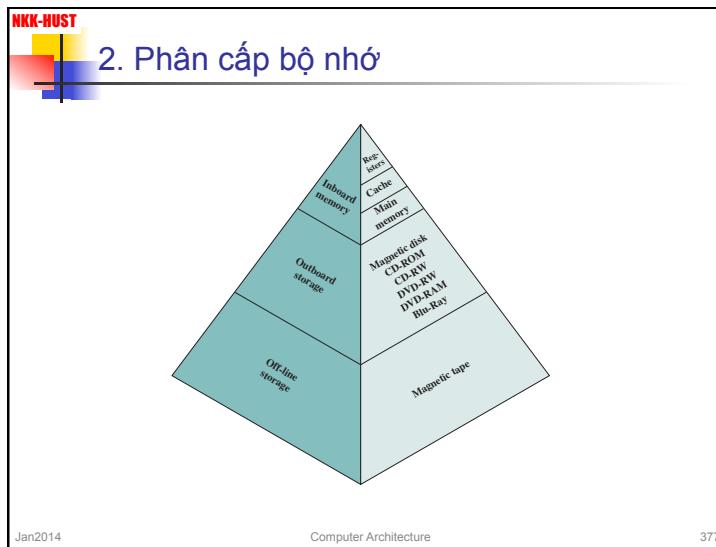
Các đặc trưng của hệ thống nhớ (tiếp)

- Các đặc tính vật lý
 - Khả biến / Không khả biến (volatile / nonvolatile)
 - Xoá được / không xoá được
- Tổ chức

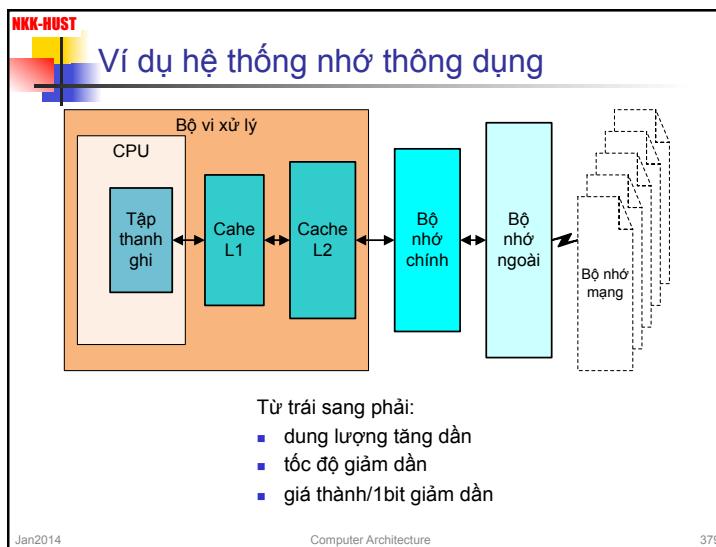
Jan2014

Computer Architecture

376



- NKK-HUST**
- ## Công nghệ bộ nhớ
- Static RAM (SRAM)
 - 0.5ns – 2.5ns, \$2000 – \$5000 per GB
 - Dynamic RAM (DRAM)
 - 50ns – 70ns, \$20 – \$75 per GB
 - Ổ đĩa từ
 - 5ms – 20ms, \$0.20 – \$2 per GB
 - Bộ nhớ lý tưởng
 - Thời gian truy nhập như SRAM
 - Dung lượng và giá thành như ổ đĩa cứng
- Jan2014 Computer Architecture 378



- NKK-HUST**
- ## Nguyên lý cục bộ hoá tham chiếu bộ nhớ
- Trong một khoảng thời gian đủ nhỏ CPU thường chỉ tham chiếu các thông tin trong một khối nhớ cục bộ
 - Ví dụ:
 - Cấu trúc chương trình tuần tự
 - Vòng lặp có thân nhỏ
 - Cấu trúc dữ liệu mảng
- Jan2014 Computer Architecture 380

NKK-HUST

7.2. Bộ nhớ chính

1. Bộ nhớ bán dẫn

Kiểu bộ nhớ	Tiêu chuẩn	Khả năng xoá	Cơ chế ghi	Tính khả biến
Read Only Memory (ROM)	Bộ nhớ chỉ đọc	Không xoá được	Mặt nạ	Không khả biến
Programmable ROM (PROM)				
Erasable PROM (EPROM)	Bộ nhớ hàm như chỉ đọc	băng tia cực tím, cá chip	Băng điện	Không khả biến
Electrically Erasable PROM (EEPROM)		băng điện, mức từng byte		
Flash memory	Bộ nhớ đọc-ghi	băng điện, từng khối	Băng điện	Khả biến
Random Access Memory (RAM)		băng điện, mức từng byte		

Jan2014 Computer Architecture 381

NKK-HUST

ROM (Read Only Memory)

- Bộ nhớ không khả biến
- Lưu trữ các thông tin sau:
 - Thư viện các chương trình con
 - Các chương trình điều khiển hệ thống (BIOS)
 - Các bảng chức năng
 - Vi chương trình

Jan2014 Computer Architecture 382

NKK-HUST

Các kiểu ROM

- ROM mặt nạ:
 - thông tin được ghi khi sản xuất
 - rất đắt
- PROM (Programmable ROM)
 - Cần thiết bị chuyên dụng để ghi bằng chương trình → chỉ ghi được một lần
- EPROM (Erasable PROM)
 - Cần thiết bị chuyên dụng để ghi bằng chương trình → ghi được nhiều lần
 - Trước khi ghi lại, xóa bằng tia cực tím

Jan2014 Computer Architecture 383

NKK-HUST

Các kiểu ROM (tiếp)

- EEPROM (Electrically Erasable PROM)
 - Có thể ghi theo từng byte
 - Xóa bằng điện
- Flash memory (Bộ nhớ cực nhanh)
 - Ghi theo khối
 - Xóa bằng điện

Jan2014 Computer Architecture 384

RAM (Random Access Memory)

- Bộ nhớ đọc-ghi (Read/Write Memory)
- Khả biến
- Lưu trữ thông tin tạm thời
- Có hai loại: SRAM và DRAM
(Static and Dynamic)

Jan2014

Computer Architecture

385

SRAM (Static) – RAM tĩnh

- Các bit được lưu trữ bằng các Flip-Flop
→ thông tin ổn định
- Cấu trúc phức tạp
- Dung lượng chip nhỏ
- Tốc độ nhanh
- Đắt tiền
- Dùng làm bộ nhớ cache

Jan2014

Computer Architecture

386

DRAM (Dynamic) – RAM động

- Các bit được lưu trữ trên tụ điện
→ cần phải có mạch làm tươi
- Cấu trúc đơn giản
- Dung lượng lớn
- Tốc độ chậm hơn
- Rẻ tiền hơn
- Dùng làm bộ nhớ chính

Jan2014

Computer Architecture

387

Các DRAM tiên tiến

- Enhanced DRAM
- Cache DRAM
- Synchronous DRAM (SDRAM): làm việc
được đồng bộ bởi xung clock
- DDR-SDRAM (Double Data Rate SDRAM)
- Rambus DRAM (RDRAM)

Jan2014

Computer Architecture

388

NKK-HUST

Tổ chức của chip nhớ

- Sơ đồ cơ bản của chip nhớ

Diagram illustrating the basic structure of a memory chip. It shows a central box labeled "Chip nhớ $2^n \times m$ bit" with multiple input and output lines. Address lines A_0, A_1, \dots, A_{n-1} enter from the left, and data lines D_0, D_1, \dots, D_{m-1} exit to the right. Control lines include \overline{CS} (Chip Select), WE (Write Enable), and \overline{OE} (Output Enable).

Jan2014 Computer Architecture 389

NKK-HUST

Các tín hiệu của chip nhớ

- Các đường địa chỉ: $A_{n-1} \div A_0 \rightarrow$ có 2^n từ nhớ
- Các đường dữ liệu: $D_{m-1} \div D_0 \rightarrow$ độ dài từ nhớ = m bit
- Dung lượng chip nhớ = $2^n \times m$ bit
- Các đường điều khiển:
 - Tín hiệu chọn chip CS (Chip Select)
 - Tín hiệu điều khiển đọc OE (Output Enable)
 - Tín hiệu điều khiển ghi WE (Write Enable)
 (Các tín hiệu điều khiển thường tích cực với mức 0)

Jan2014 Computer Architecture 390

NKK-HUST

Tổ chức của DRAM

- Dùng n đường địa chỉ dòn kẽnh \rightarrow cho phép truyền $2n$ bit địa chỉ
- Tín hiệu chọn địa chỉ hàng RAS (Row Address Select)
- Tín hiệu chọn địa chỉ cột CAS (Column Address Select)
- Dung lượng của DRAM = $2^{2n} \times m$ bit

Jan2014 Computer Architecture 391

NKK-HUST

Ví dụ chip nhớ

Diagram showing pinouts for two types of memory chips:

- (a) 8-Mbit EPROM: 32-pin DIP package. Pins 1-8 are address lines (A19-A12), pins 9-16 are address lines (A11-A8), pin 17 is Vcc, pins 18-24 are data lines (D0-D6), pin 25 is WE, pin 26 is NC, pin 27 is RAS, pin 28 is CAS, pin 29 is OE, and pins 30-32 are A17-A18.
- (b) 16-Mbit DRAM: 24-pin DIP package. Pins 1-4 are address lines (A19-A16), pins 5-8 are address lines (A15-A12), pin 9 is Vcc, pins 10-13 are address lines (A11-A8), pin 14 is A10, pin 15 is CE, pin 16 is D7, pin 17 is D6, pin 18 is D5, pin 19 is D4, pin 20 is D3, pin 21 is D2, pin 22 is D1, pin 23 is D0, pin 24 is Vss, and pins 25-28 are A9-A6.

Jan2014 Computer Architecture 392



Thiết kế mô-đun nhớ bán dẫn

- Dung lượng chip nhớ $2^n \times m$ bit
- Cần thiết kế để tăng dung lượng:
 - Thiết kế tăng độ dài từ nhớ
 - Thiết kế tăng số lượng từ nhớ
 - Thiết kế kết hợp

Jan2014

Computer Architecture

393



Tăng độ dài từ nhớ

VD1:

- Cho chip nhớ SRAM $4K \times 4$ bit
- Thiết kế mô-đun nhớ $4K \times 8$ bit

Giải:

- Dung lượng chip nhớ = $2^{12} \times 4$ bit
- chip nhớ có:
 - 12 chân địa chỉ
 - 4 chân dữ liệu
- mô-đun nhớ cần có:
 - 12 chân địa chỉ
 - 8 chân dữ liệu

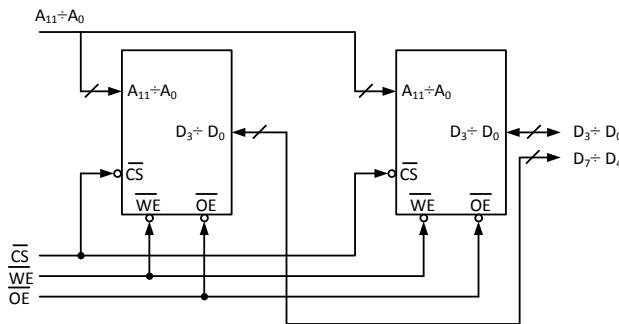
Jan2014

Computer Architecture

394



Ví dụ tăng độ dài từ nhớ



Jan2014

Computer Architecture

395



Bài toán tăng độ dài từ nhớ tổng quát

- Cho chip nhớ $2^n \times m$ bit
- Thiết kế mô-đun nhớ $2^n \times (k.m)$ bit
- Dùng k chip nhớ

Jan2014

Computer Architecture

396

NKK-HUST

Tăng số lượng từ nhớ

VD2:

- Cho chip nhớ SRAM $4K \times 8$ bit
- Thiết kế mô-đun nhớ $8K \times 8$ bit

Giải:

- Dung lượng chip nhớ = $2^{12} \times 8$ bit
- chip nhớ có:
 - 12 chân địa chỉ
 - 8 chân dữ liệu
- Dung lượng mô-đun nhớ = $2^{13} \times 8$ bit
 - 13 chân địa chỉ
 - 8 chân dữ liệu

Jan2014 Computer Architecture 397

NKK-HUST

Tăng số lượng từ nhớ

\bar{G}	A	\bar{Y}_0	\bar{Y}_1
0	0	0	1
0	1	1	0
1	x	1	1

Jan2014 Computer Architecture 398

NKK-HUST

Bài tập

- Tăng số lượng từ gấp 4 lần:
 - Cho chip nhớ SRAM $4K \times 8$ bit
 - Thiết kế mô-đun nhớ $16K \times 8$ bit
- Tăng số lượng từ gấp 8 lần:
 - Cho chip nhớ SRAM $4K \times 8$ bit
 - Thiết kế mô-đun nhớ $32K \times 8$ bit
- Thiết kế kết hợp:
 - Cho chip nhớ SRAM $4K \times 4$ bit
 - Thiết kế mô-đun nhớ $8K \times 8$ bit

Jan2014 Computer Architecture 399

NKK-HUST

Bộ giải mã $2 \rightarrow 4$

\bar{G}	B	A	\bar{Y}_0	\bar{Y}_1	\bar{Y}_2	\bar{Y}_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	x	x	1	1	1	1

Jan2014 Computer Architecture 400

Bài tập

- Tăng số lượng từ nhớ gấp 3 lần
- Tăng số lượng từ nhớ gấp 5, 6, 7 lần

Jan2014

Computer Architecture

401

2. Các đặc trưng cơ bản của bộ nhớ chính

- Chứa các chương trình đang thực hiện và các dữ liệu đang được sử dụng
- Tồn tại trên mọi hệ thống máy tính
- Bao gồm các ngăn nhớ được đánh địa chỉ trực tiếp bởi CPU
- Dung lượng của bộ nhớ chính nhỏ hơn không gian địa chỉ bộ nhớ mà CPU quản lý.
- Việc quản lý logic bộ nhớ chính tuỳ thuộc vào hệ điều hành

Jan2014

Computer Architecture

402

Tổ chức bộ nhớ đan xen (interleaved memory)

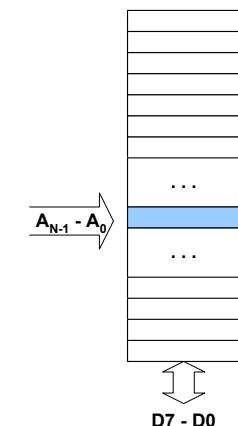
- Độ rộng của bus dữ liệu để trao đổi với bộ nhớ: $m = 8, 16, 32, 64, 128 \dots$ bit
- Các ngăn nhớ được tổ chức theo byte
→ tổ chức bộ nhớ vật lý khác nhau

Jan2014

Computer Architecture

403

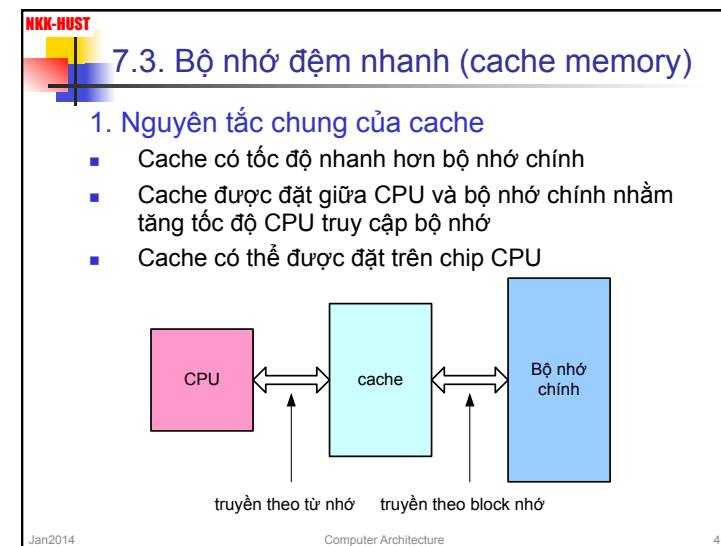
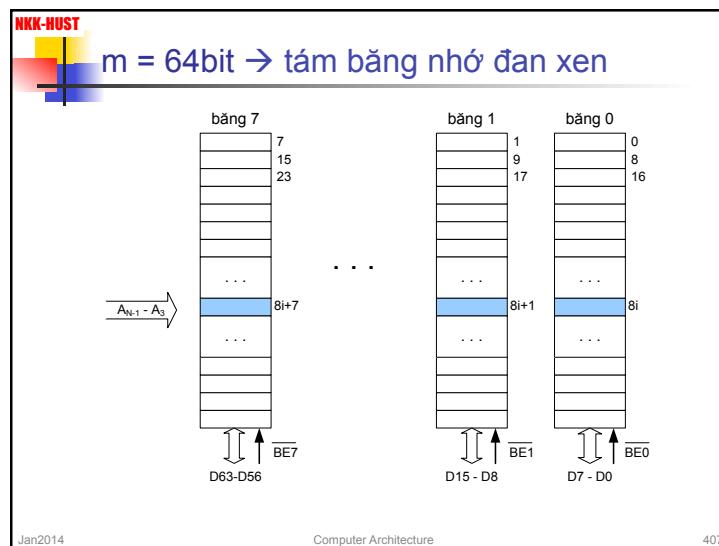
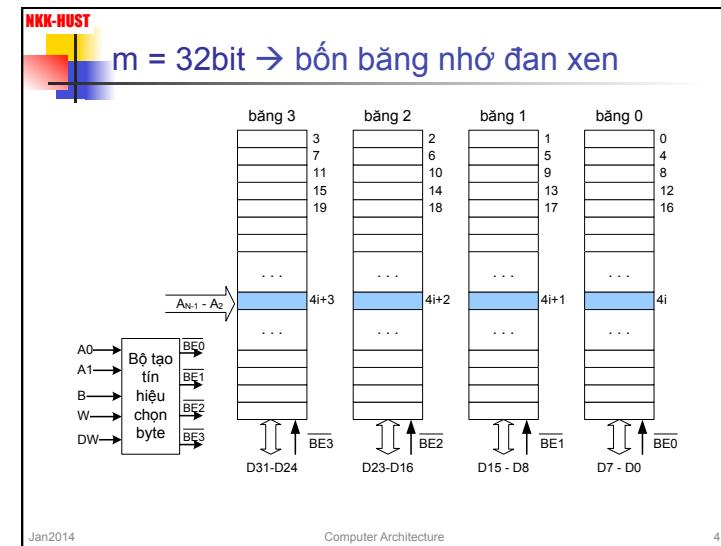
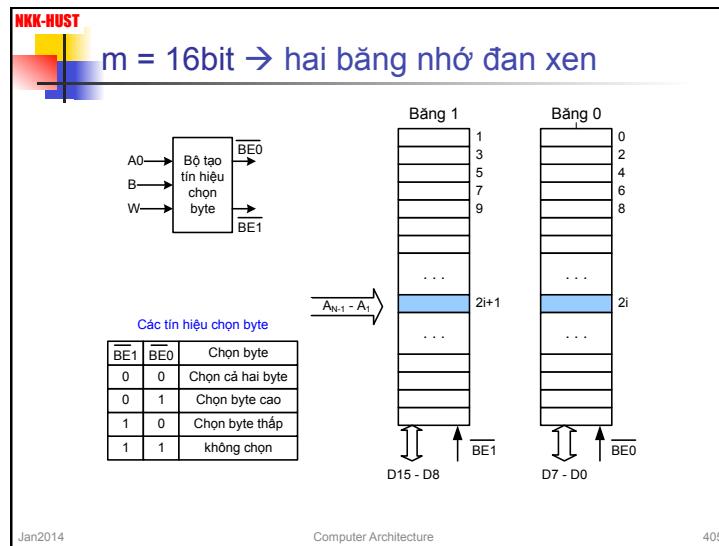
$m=8\text{bit} \rightarrow$ một băng nhớ tuyến tính



Jan2014

Computer Architecture

404



Ví dụ về thao tác của cache

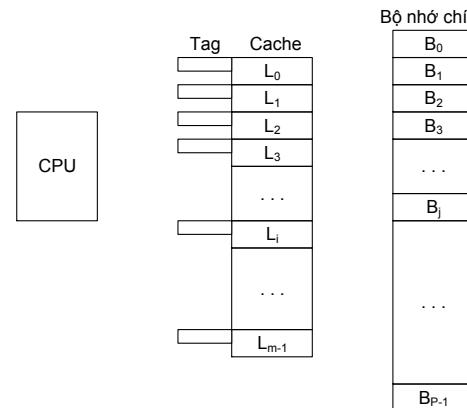
- CPU yêu cầu nội dung của ngăn nhớ
- CPU kiểm tra trên cache với dữ liệu này
- Nếu có, CPU nhận dữ liệu từ cache (nhanh)
- Nếu không có, đọc Block nhớ chứa dữ liệu từ bộ nhớ chính vào cache
- Tiếp đó chuyển dữ liệu từ cache vào CPU

Jan2014

Computer Architecture

409

Cấu trúc chung của cache / bộ nhớ chính



Jan2014

Computer Architecture

410

Cấu trúc chung của cache / bộ nhớ chính (tiếp)

- Bộ nhớ chính có 2^N byte nhớ
- Bộ nhớ chính và cache được chia thành các khối có kích thước bằng nhau
 - Bộ nhớ chính: $B_0, B_1, B_2, \dots, B_{p-1}$ (p Blocks)
 - Bộ nhớ cache: $L_0, L_1, L_2, \dots, L_{m-1}$ (m Lines)
 - Kích thước của Block = 8,16,32,64,128 byte

Jan2014

Computer Architecture

411

Cấu trúc chung của cache / bộ nhớ chính (tiếp)

- Một số Block của bộ nhớ chính được nạp vào các Line của cache.
- Nội dung Tag (thẻ nhớ) cho biết Block nào của bộ nhớ chính hiện đang được chứa ở Line đó.
- Khi CPU truy nhập (đọc/ghi) một từ nhớ, có hai khả năng xảy ra:
 - Từ nhớ đó có trong cache (cache hit)
 - Từ nhớ đó không có trong cache (cache miss).

Jan2014

Computer Architecture

412

NKK-HUST

2. Các phương pháp ánh xạ

(Chính là các phương pháp tổ chức bộ nhớ cache)

- Ánh xạ trực tiếp
(Direct mapping)
- Ánh xạ liên kết toàn phần
(Fully associative mapping)
- Ánh xạ liên kết tập hợp
(Set associative mapping)

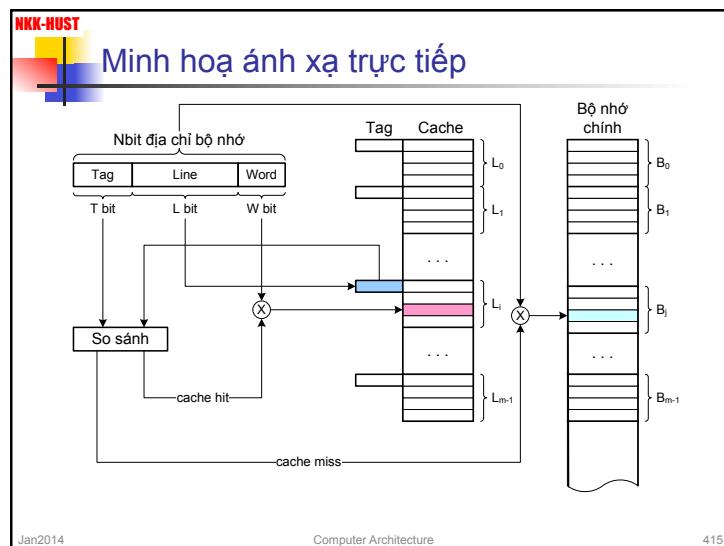
Jan2014 Computer Architecture 413

NKK-HUST

Ánh xạ trực tiếp

- Mỗi Block của bộ nhớ chính chỉ có thể được nạp vào một Line của cache:
 - $B_0 \rightarrow L_0$
 - $B_1 \rightarrow L_1$
 -
 - $B_{m-1} \rightarrow L_{m-1}$
 - $B_m \rightarrow L_0$
 - $B_{m+1} \rightarrow L_1$
 -
- Tổng quát
 - B_j chỉ có thể nạp vào $L_{j \bmod m}$
 - m là số Line của cache.

Jan2014 Computer Architecture 414



NKK-HUST

Đặc điểm của ánh xạ trực tiếp

- Mỗi một địa chỉ N bit của bộ nhớ chính gồm ba trường:
 - **Trường Word** gồm W bit xác định một từ nhớ trong Block hay Line:
 $2^W =$ kích thước của Block hay Line
 - **Trường Line** gồm L bit xác định một trong số các Line trong cache:
 $2^L =$ số Line trong cache = m
 - **Trường Tag** gồm T bit:
 $T = N - (W+L)$
- Bộ so sánh đơn giản
- Xác suất cache hit thấp

Jan2014 Computer Architecture 416

Ánh xạ liên kết toàn phần

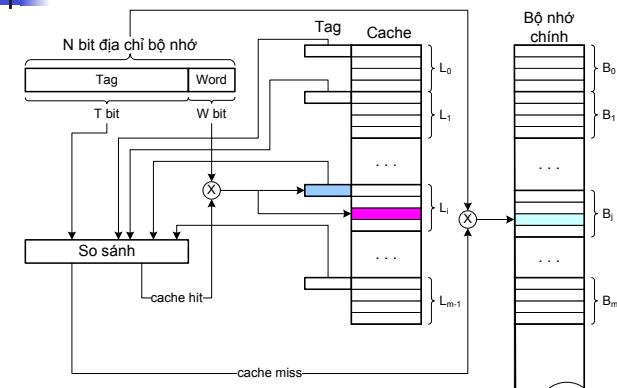
- Mỗi *Block* có thể nạp vào bất kỳ *Line* nào của *cache*.
- Địa chỉ của bộ nhớ chính bao gồm hai trường:
 - Trường *Word* giống như trường hợp ở trên.
 - Trường *Tag* dùng để xác định *Block* của bộ nhớ chính.
- Tag xác định *Block* đang nằm ở *Line* đó

Jan2014

Computer Architecture

417

Minh họa ánh xạ liên kết toàn phần



Jan2014

Computer Architecture

418

Đặc điểm của ánh xạ liên kết toàn phần

- So sánh đồng thời với tất cả các Tag → mất nhiều thời gian
- Xác suất *cache hit* cao.
- Bộ so sánh phức tạp.

Jan2014

Computer Architecture

419

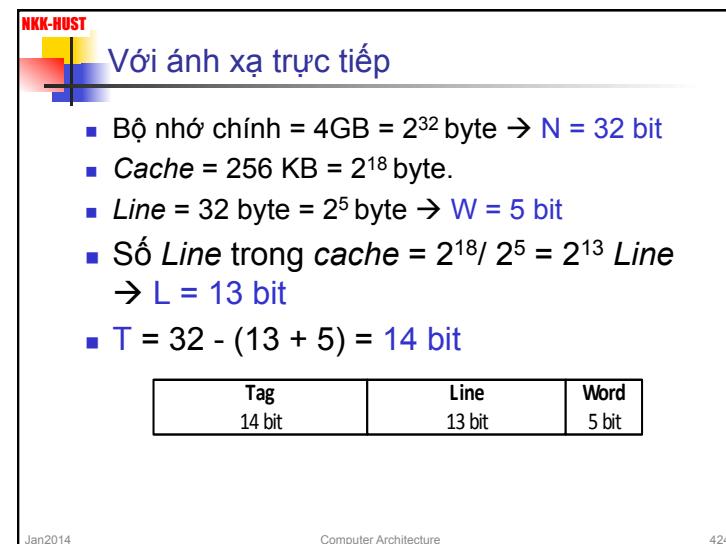
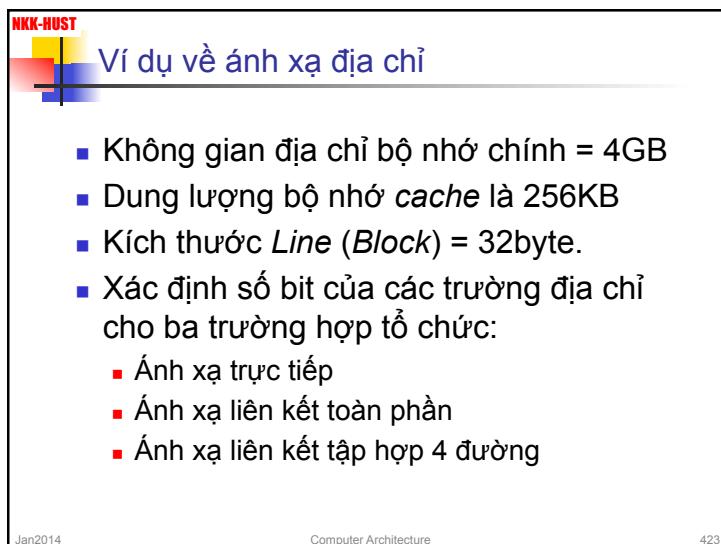
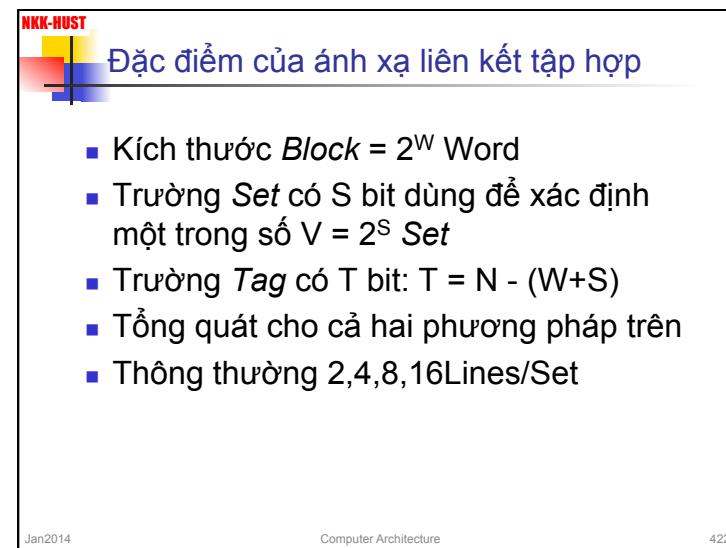
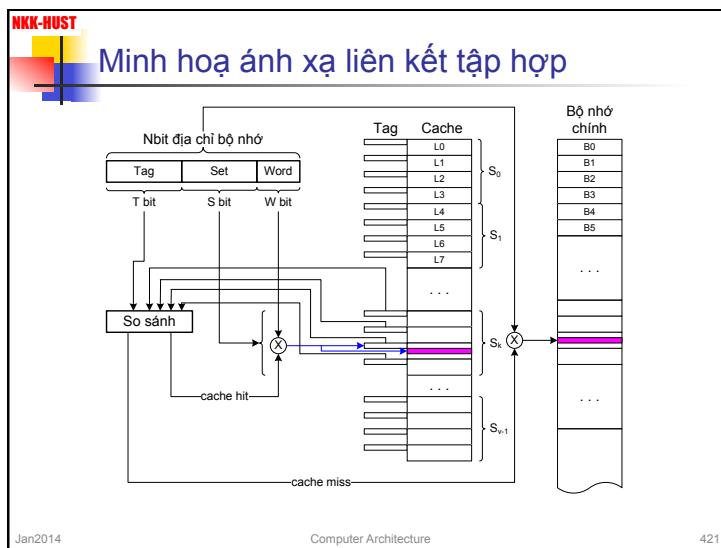
Ánh xạ liên kết tập hợp

- Cache được chia thành các Tập (Set)
- Mỗi một Set chứa một số Line
- Ví dụ:
 - 4 Line/Set → 4-way associative mapping
- Ánh xạ theo nguyên tắc sau:
 - B₀ → S₀
 - B₁ → S₁
 - B₂ → S₂
 -

Jan2014

Computer Architecture

420



NKK-HUST

Với ánh xạ liên kết toàn phần

- Bộ nhớ chính = 4GB = 2^{32} byte → N = 32 bit
- Line = 32 byte = 2^5 byte → W = 5 bit
- Số bit của trường Tag sẽ là: T = 32 - 5 = 27 bit

Tag 27 bit	Word 5 bit
---------------	---------------

Jan2014 Computer Architecture 425

NKK-HUST

Với ánh xạ liên kết tập hợp 4 đường

- Bộ nhớ chính = 4GB = 2^{32} byte → N = 32 bit
- Line = 32 byte = 2^5 byte → W = 5 bit
- Số Line trong cache = $2^{18} / 2^5 = 2^{13}$ Line
- Một Set có 4 Line = 2^2 Line
- số Set trong cache = $2^{13} / 2^2 = 2^{11}$ Set → S = 11 bit
- Số bit của trường Tag sẽ là: T = 32 - (11 + 5) = 16 bit

Tag 16 bit	Set 11 bit	Word 5 bit
---------------	---------------	---------------

Jan2014 Computer Architecture 426

NKK-HUST

Bài tập

Giả thiết rằng máy tính có 128KB cache tổ chức theo kiểu ánh xạ liên kết tập hợp 4-line. Cache có tất cả là 1024 Set từ S0 đến S1023. Địa chỉ bộ nhớ chính là 32-bit và đánh địa chỉ cho từng byte.

a) Tính số bit cho các trường địa chỉ khi truy nhập cache ?

b) Xác định byte nhớ có địa chỉ 003D02AF₍₁₆₎ được ánh xạ vào Set nào của cache ?

Jan2014 Computer Architecture 427

NKK-HUST

3. Thay thế block trong cache

Với ánh xạ trực tiếp:

- Không phải lựa chọn
- Mỗi Block chỉ ánh xạ vào một Line xác định
- Thay thế Block ở Line đó

Jan2014 Computer Architecture 428

Thay thế block trong cache (tiếp)

Với ánh xạ liên kết: cần có thuật giải thay thế:

- **Random**: Thay thế ngẫu nhiên
- **FIFO** (First In First Out): Thay thế *Block* nào nằm lâu nhất ở trong Set đó
- **LFU** (Least Frequently Used): Thay thế *Block* nào trong Set có số lần truy nhập ít nhất trong cùng một khoảng thời gian
- **LRU** (Least Recently Used): Thay thế *Block* ở trong Set tương ứng có thời gian lâu nhất không được tham chiếu tới.
- **Tối ưu nhất: LRU**

Jan2014

Computer Architecture

429

4. Phương pháp ghi dữ liệu khi cache hit

■ Ghi xuyên qua (Write-through):

- ghi cả cache và cả bộ nhớ chính
- tốc độ chậm

■ Ghi trả sau (Write-back):

- chỉ ghi ra cache
- tốc độ nhanh
- khi Block trong cache bị thay thế cần phải ghi trả cả Block về bộ nhớ chính

Jan2014

Computer Architecture

430

7.4. Bộ nhớ ngoài

Các kiểu bộ nhớ ngoài

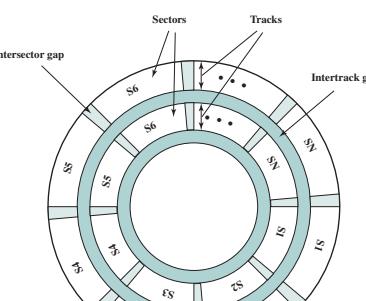
- Băng từ
- Đĩa từ
- Đĩa quang
- Bộ nhớ Flash

Jan2014

Computer Architecture

431

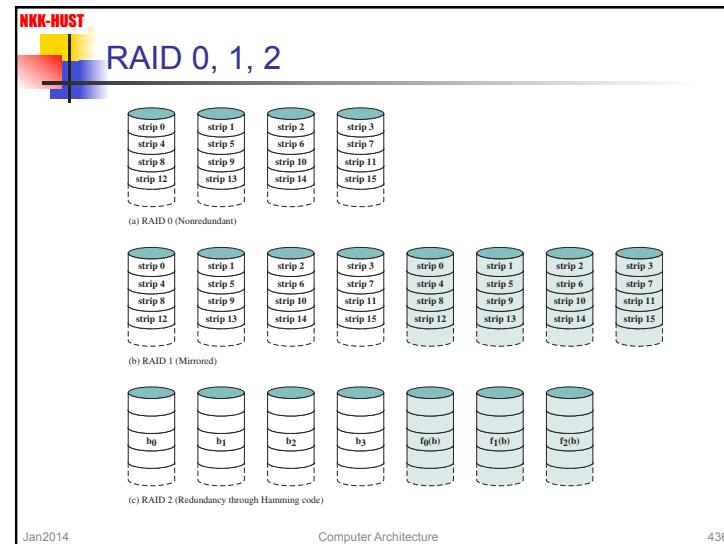
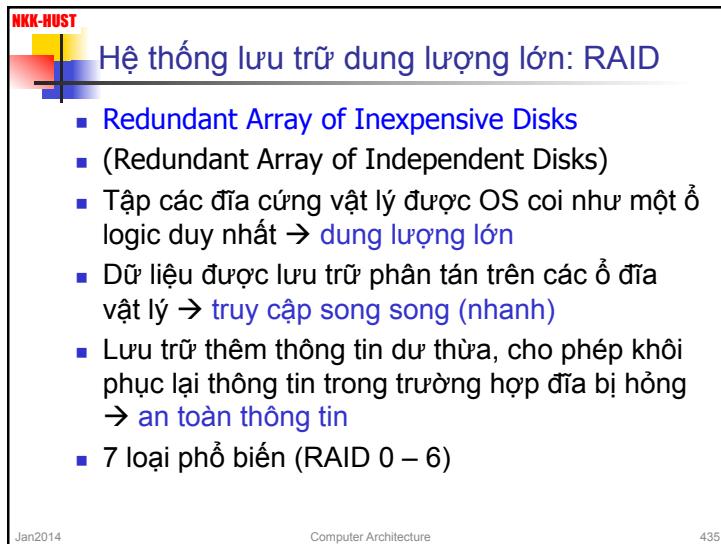
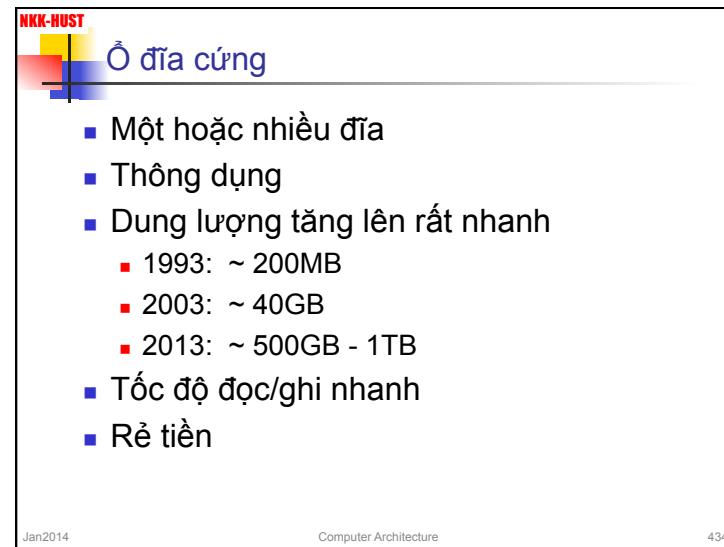
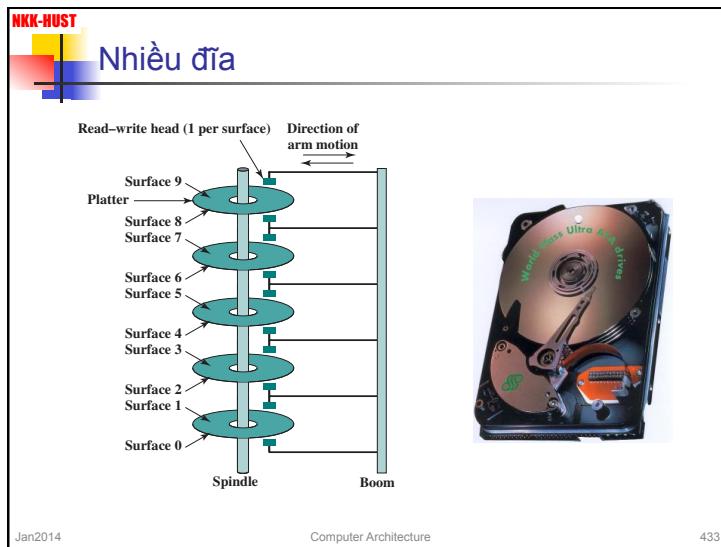
Đĩa từ

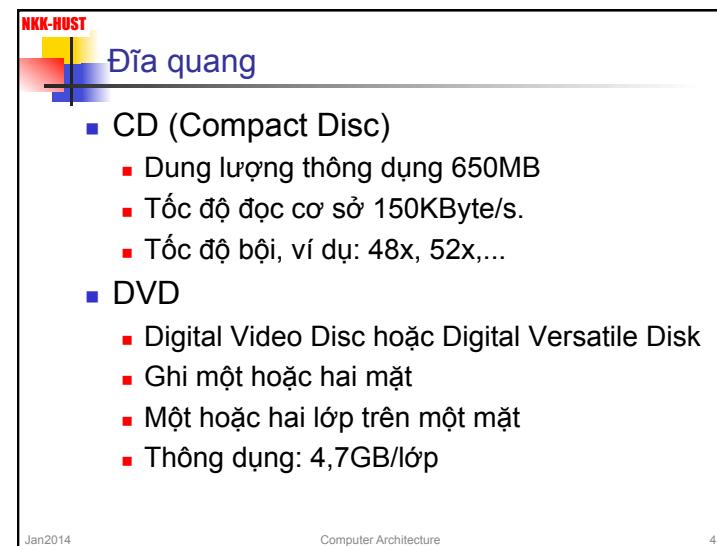
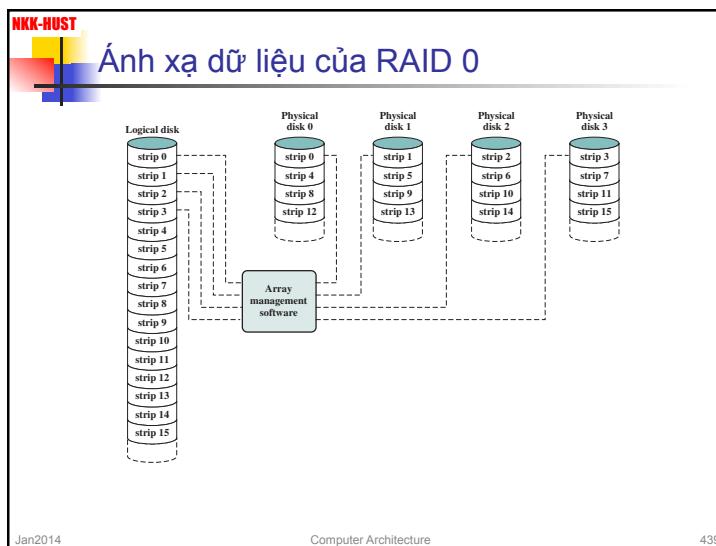
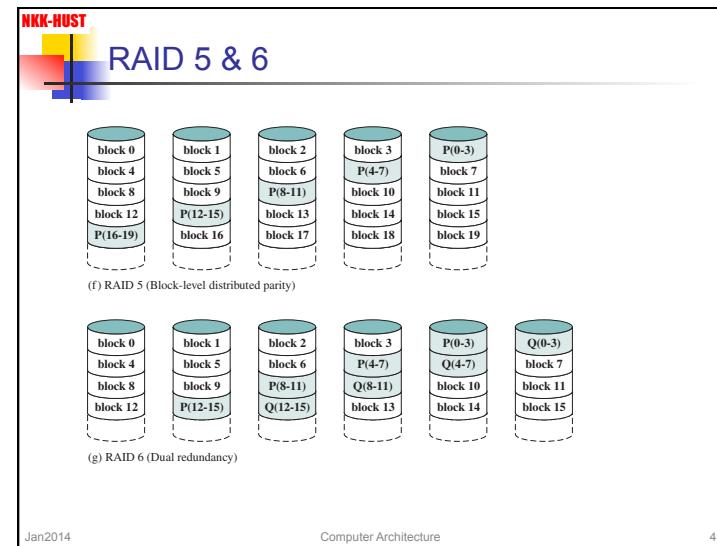
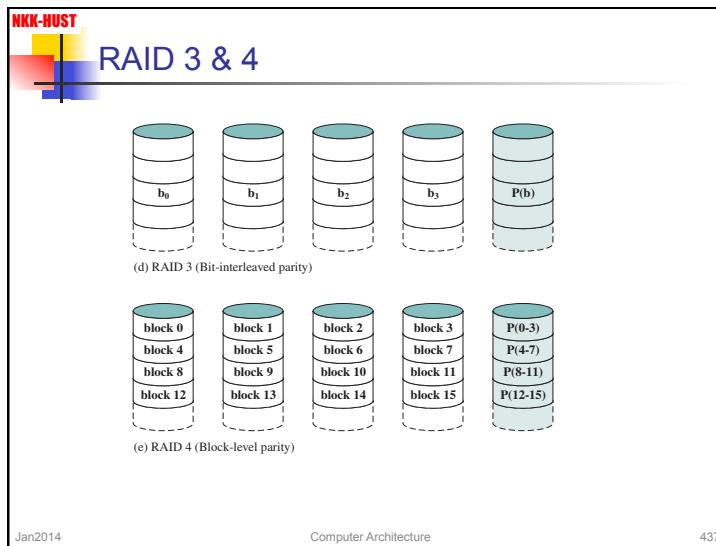


Jan2014

Computer Architecture

432





NKK-HUST

Bộ nhớ flash

- Bộ nhớ bán dẫn
- Không khả biến
- Tốc độ nhanh
- Các dạng:
 - Ổ nhớ kết nối qua cổng USB
 - Thẻ nhớ
 - Ổ SSD (Solid State Drive): kết nối nhiều chip nhớ flash và cho phép truy cập song song



Jan2014 Computer Architecture 441

NKK-HUST

7.5. Bộ nhớ ảo (Virtual Memory)

- Khái niệm bộ nhớ ảo: gồm bộ nhớ chính và bộ nhớ ngoài mà được CPU coi như là một bộ nhớ duy nhất (bộ nhớ chính).
- Các kỹ thuật thực hiện bộ nhớ ảo:
 - Kỹ thuật phân trang: Chia không gian địa chỉ bộ nhớ thành các trang nhớ có kích thước bằng nhau và nằm liền kề nhau
Thông dụng: kích thước trang = 4KBytes
 - Kỹ thuật phân đoạn: Chia không gian nhớ thành các đoạn nhớ có kích thước thay đổi, các đoạn nhớ có thể gói lên nhau.

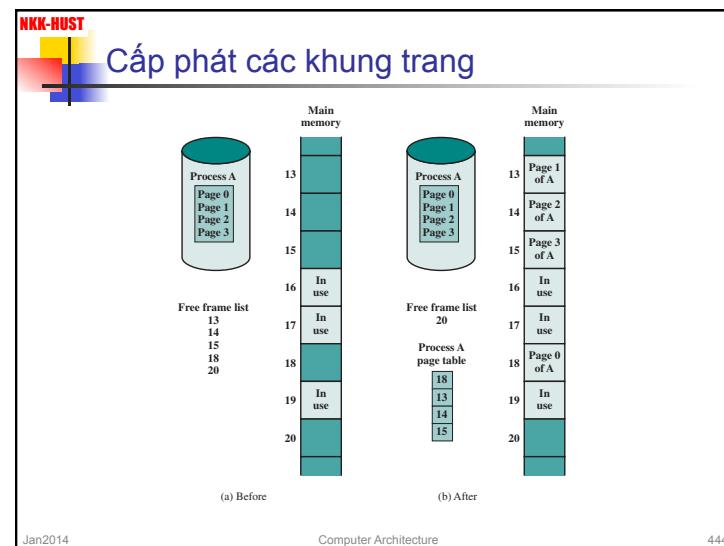
Jan2014 Computer Architecture 442

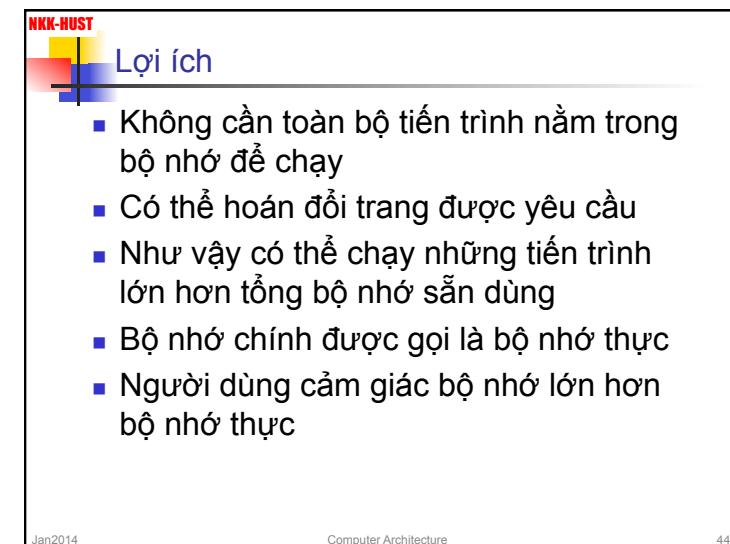
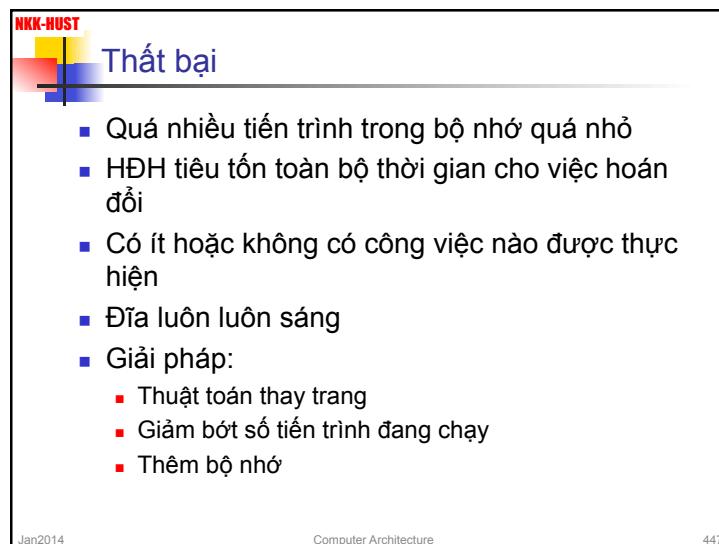
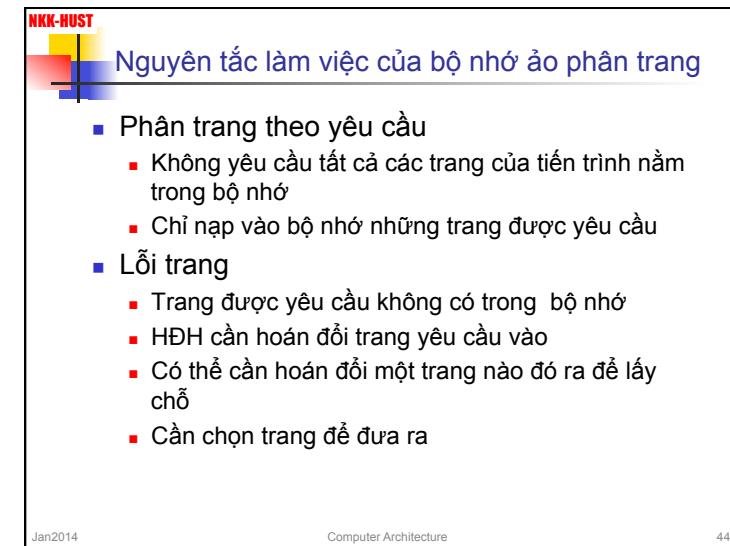
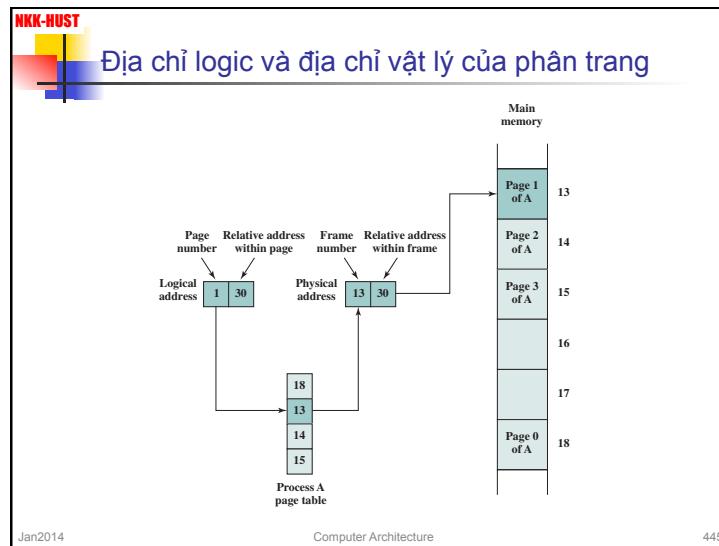
NKK-HUST

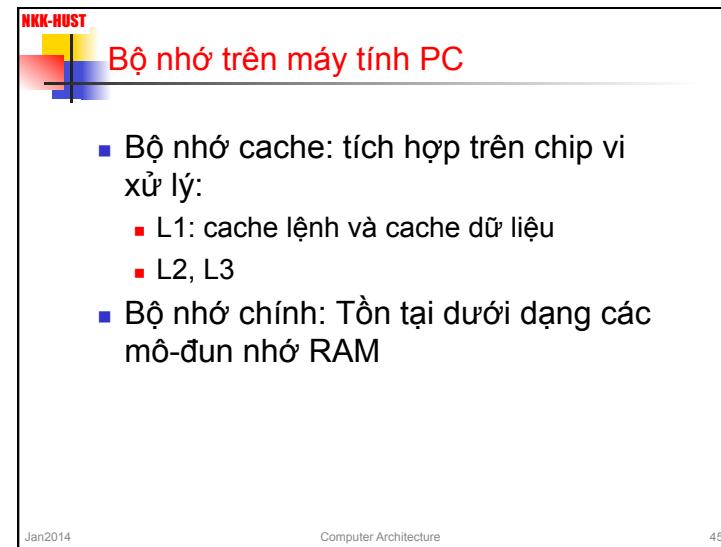
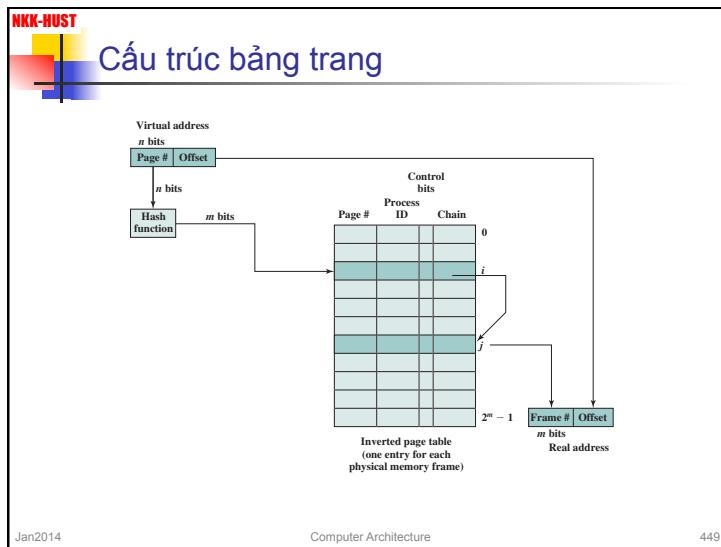
Phân trang

- Phân chia bộ nhớ thành các phần có kích thước bằng nhau gọi là các khung trang
- Chia chương trình (tiến trình) thành các trang
- Cấp phát số hiệu khung trang yêu cầu cho tiến trình
- HĐH duy trì danh sách các khung trang nhớ trống
- Tiến trình không yêu cầu các khung trang liên tiếp
- Sử dụng bảng trang để quản lý

Jan2014 Computer Architecture 443







- Bộ nhớ trên PC (tiếp)**
- ROM BIOS chứa các chương trình sau:
 - Chương trình POST (Power On Self Test)
 - Chương trình CMOS Setup
 - Chương trình Bootstrap loader
 - Các trình điều khiển vào-ra cơ bản (BIOS)
 - CMOS RAM:
 - Chứa thông tin cấu hình hệ thống
 - Đồng hồ hệ thống
 - Có pin nuôi riêng
 - Video RAM: quản lý thông tin của màn hình
 - Các loại bộ nhớ ngoài
- Jan2014 Computer Architecture 451

Hết chương 7

Jan2014 Computer Architecture 452

NKK-HUST

Kiến trúc máy tính

Chương 8 HỆ THỐNG VÀO-RA

Nguyễn Kim Khánh
Trường Đại học Bách khoa Hà Nội

Jan2014 Computer Architecture 453

NKK-HUST

Nội dung học phần

- Chương 1. Giới thiệu chung
- Chương 2. Cơ bản về logic số
- Chương 3. Hệ thống máy tính
- Chương 4. Số học máy tính
- Chương 5. Kiến trúc tập lệnh
- Chương 6. Bộ xử lý
- Chương 7. Bộ nhớ máy tính
- Chương 8. Hệ thống vào-ra**
- Chương 9. Các kiến trúc song song

Jan2014 Computer Architecture 454

NKK-HUST

Nội dung của chương 8

- 8.1. Tổng quan về hệ thống vào-ra
- 8.2. Các phương pháp điều khiển vào-ra
- 8.3. Nối ghép thiết bị ngoại vi

Jan2014 Computer Architecture 455

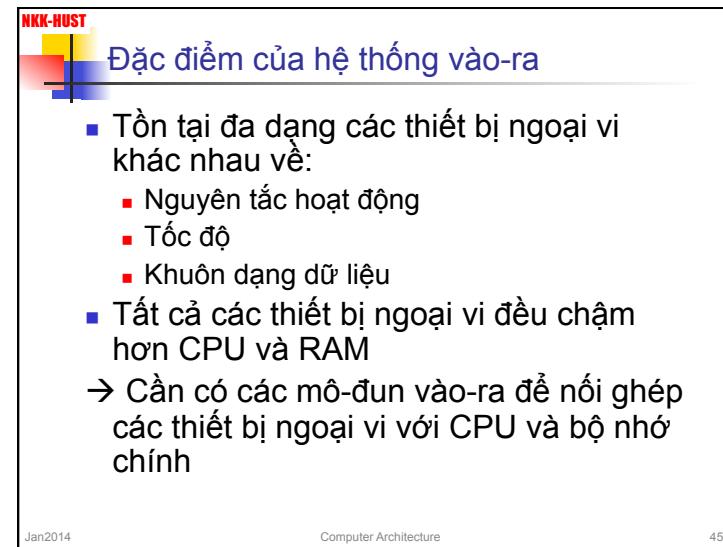
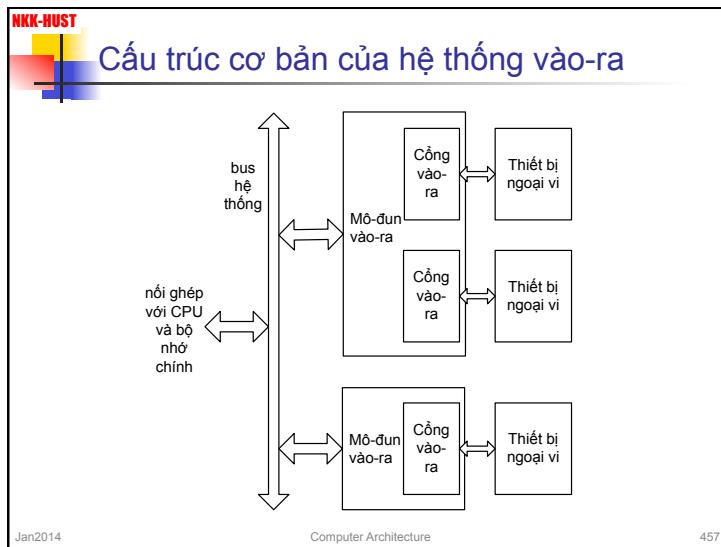
NKK-HUST

8.1. Tổng quan về hệ thống vào-ra

1. Giới thiệu chung

- Chức năng của hệ thống vào-ra: Trao đổi thông tin giữa máy tính với thế giới bên ngoài
- Các thao tác cơ bản:
 - Vào dữ liệu (Input)
 - Ra dữ liệu (Output)
- Các thành phần chính:
 - Các thiết bị ngoại vi
 - Các mô-đun vào-ra

Jan2014 Computer Architecture 456

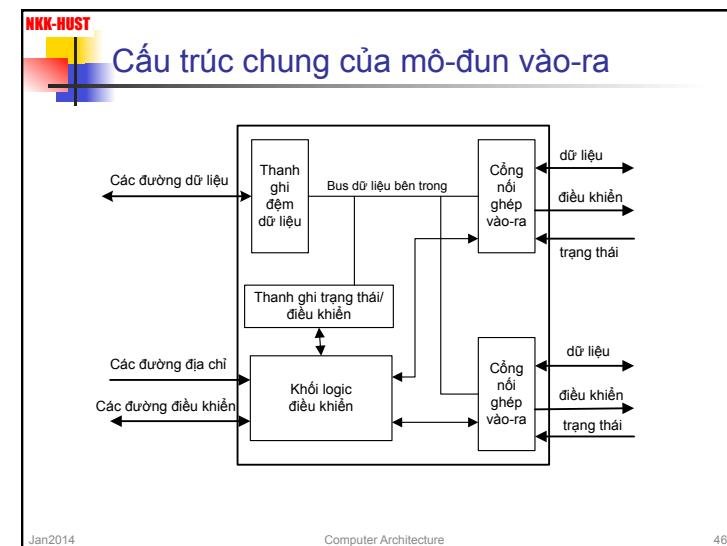
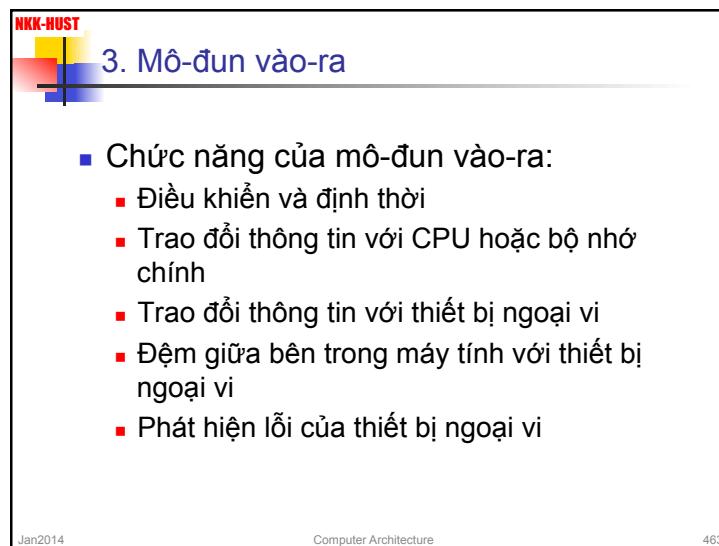
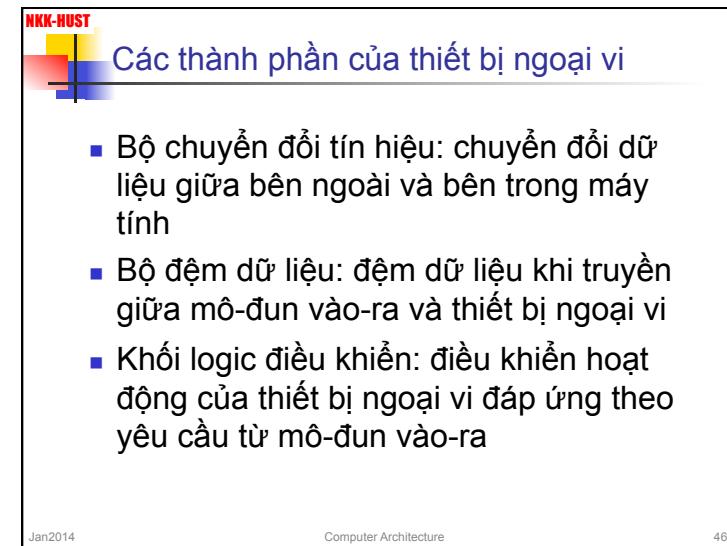
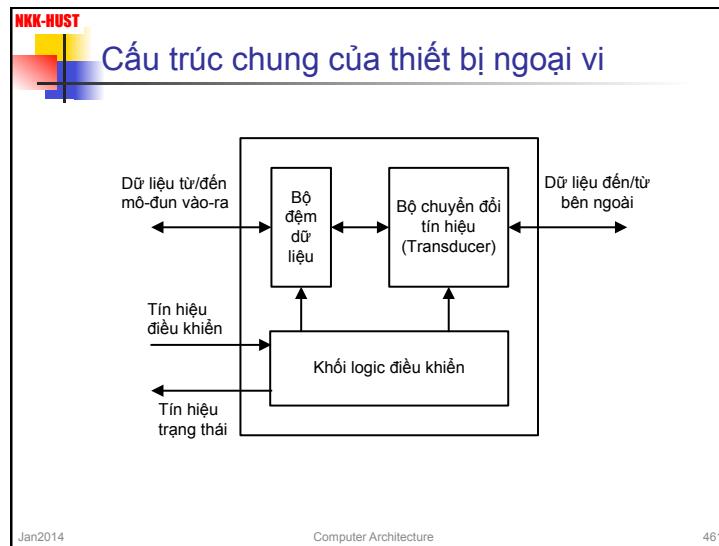


- 2. Các thiết bị ngoại vi**
- Chức năng: chuyển đổi dữ liệu giữa bên trong và bên ngoài máy tính
 - Phân loại:
 - Thiết bị ngoại vi giao tiếp người-máy: Bàn phím, Màn hình, Máy in,...
 - Thiết bị ngoại vi giao tiếp máy-máy: gồm các thiết bị theo dõi và kiểm tra
 - Thiết bị ngoại vi truyền thông: Modem, Network Interface Card (NIC)
- Jan2014 Computer Architecture 459

Một số thiết bị ngoại vi

Device	Behavior	Partner	Data rate (Mbit/sec)
Keyboard	Input	Human	0.0001
Mouse	Input	Human	0.0038
Voice input	Input	Human	0.2640
Sound input	Input	Machine	3.0000
Scanner	Input	Human	3.2000
Voice output	Output	Human	0.2640
Sound output	Output	Human	8.0000
Laser printer	Output	Human	3.2000
Graphics display	Output	Human	800.0000-8000.0000
Cable modem	Input or output	Machine	0.1280-6.0000
Network/LAN	Input or output	Machine	100.0000-10000.0000
Network/wireless LAN	Input or output	Machine	11.0000-54.0000
Optical disk	Storage	Machine	80.0000-220.0000
Magnetic tape	Storage	Machine	5.0000-120.0000
Flash memory	Storage	Machine	32.0000-200.0000
Magnetic disk	Storage	Machine	800.0000-3000.0000

Jan2014 Computer Architecture 460



NKK-HUST

Các thành phần của mô-đun vào-ra

- Thanh ghi đệm dữ liệu: đệm dữ liệu trong quá trình trao đổi
- Các cổng vào-ra (I/O Port): kết nối với thiết bị ngoại vi, mỗi cổng có một địa chỉ xác định
- Thanh ghi trạng thái/điều khiển: lưu giữ thông tin trạng thái/điều khiển cho các cổng vào-ra
- Khối logic điều khiển: điều khiển mô-đun vào-ra

Jan2014 Computer Architecture 465

NKK-HUST

4. Địa chỉ hóa cổng vào-ra

Không gian địa chỉ của bộ xử lý

Không gian địa chỉ bộ nhớ	N bit	Không gian địa chỉ vào-ra	N ₁ bit
000...000		00..00	
000...001		00..01	
000...010		00..10	
000...011		00..11	
000...100		.	
000...101		.	
.		.	
.		.	
.		.	
111...111		11..11	

Jan2014 Computer Architecture 466

NKK-HUST

Không gian địa chỉ của bộ xử lý (tiếp)

- Một số bộ xử lý chỉ quản lý duy nhất một không gian địa chỉ:
 - không gian địa chỉ bộ nhớ: 2^N địa chỉ
- Ví dụ:
 - Các bộ xử lý 680x0 (Motorola)
 - Các bộ xử lý theo kiến trúc RISC: MIPS, ...

Jan2014 Computer Architecture 467

NKK-HUST

Không gian địa chỉ của bộ xử lý (tiếp)

- Một số bộ xử lý quản lý hai không gian địa chỉ tách biệt:
 - Không gian địa chỉ bộ nhớ: 2^N địa chỉ
 - Không gian địa chỉ vào-ra: 2^{N_1} địa chỉ
 - Có tín hiệu điều khiển phân biệt phân truy nhập không gian địa chỉ
 - Tập lệnh có các lệnh vào-ra chuyên dụng
- Ví dụ: Pentium (Intel)
 - không gian địa chỉ bộ nhớ = 2^{32} byte = 4GB
 - không gian địa chỉ vào-ra = 2^{16} byte = 64KB
 - Tín hiệu điều khiển M/I/O
 - Lệnh vào-ra chuyên dụng: IN, OUT

Jan2014 Computer Architecture 468

Các phương pháp địa chỉ hóa cổng vào-ra

- Vào-ra riêng biệt
(Isolated IO hay IO mapped IO)
- Vào-ra theo bản đồ bộ nhớ
(Memory mapped IO)

Jan2014

Computer Architecture

469

Vào-ra riêng biệt

- Cổng vào-ra được đánh địa chỉ theo không gian địa chỉ vào-ra
- CPU trao đổi dữ liệu với cổng vào-ra thông qua các lệnh vào-ra chuyên dụng (IN, OUT)
- Chỉ có thể thực hiện trên các hệ thống có quản lý không gian địa chỉ vào-ra riêng biệt

Jan2014

Computer Architecture

470

Vào-ra theo bản đồ bộ nhớ

- Cổng vào-ra được đánh địa chỉ theo không gian địa chỉ bộ nhớ
- Vào-ra giống như đọc/ghi bộ nhớ
- CPU trao đổi dữ liệu với cổng vào-ra thông qua các lệnh truy nhập dữ liệu bộ nhớ
- Có thể thực hiện trên mọi hệ thống

Jan2014

Computer Architecture

471

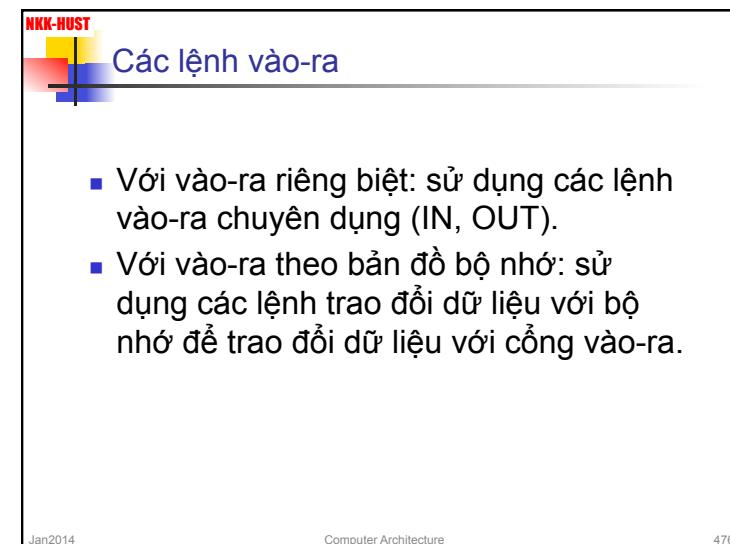
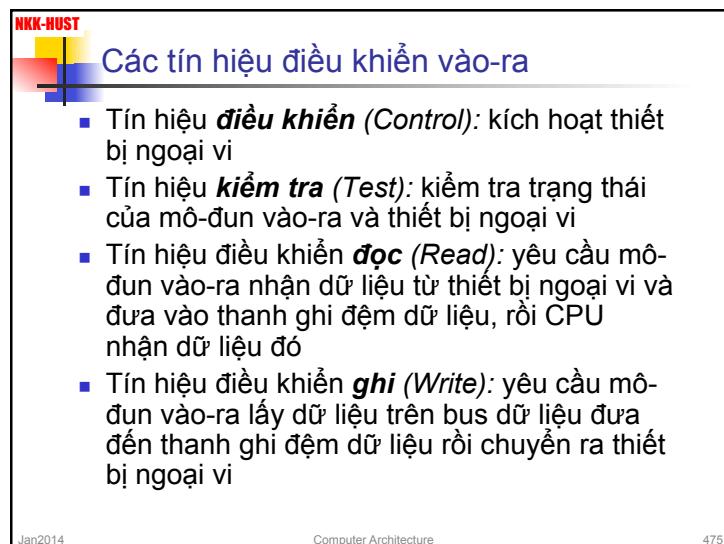
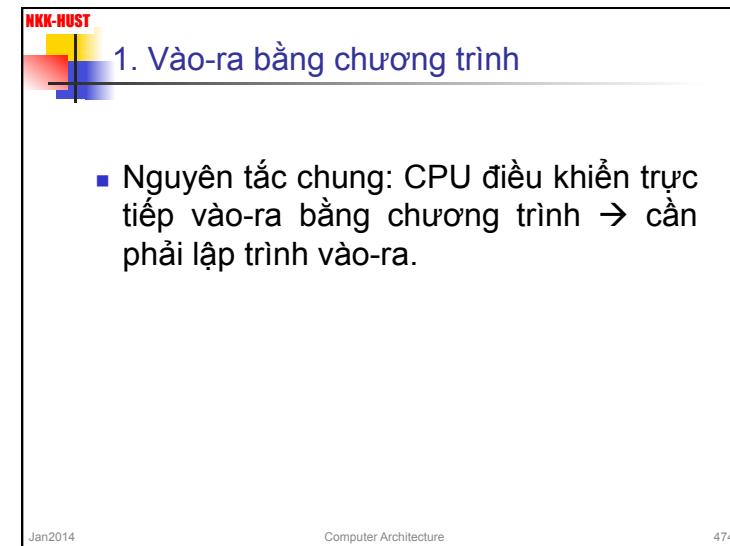
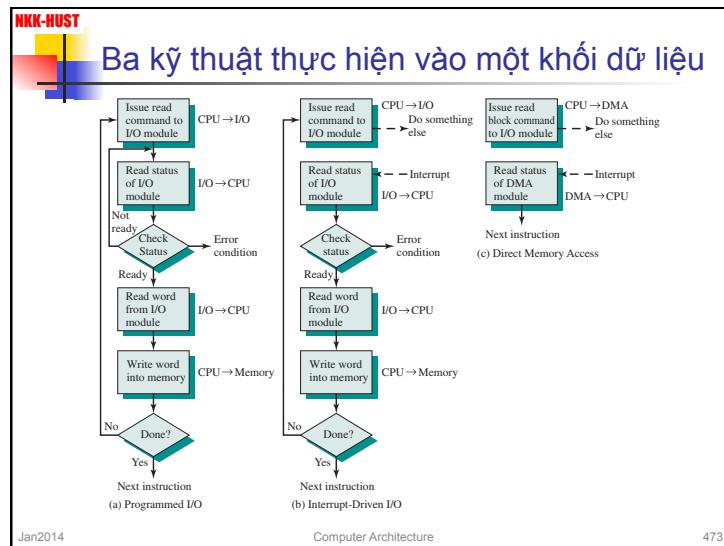
8.2. Các phương pháp điều khiển vào-ra

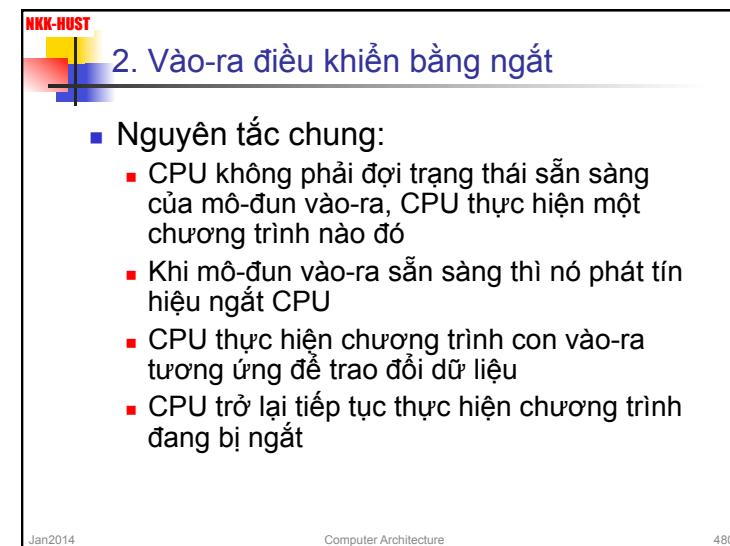
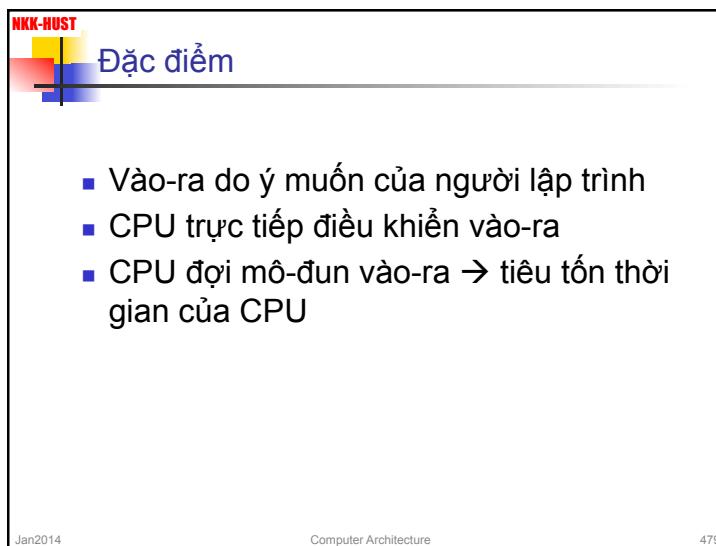
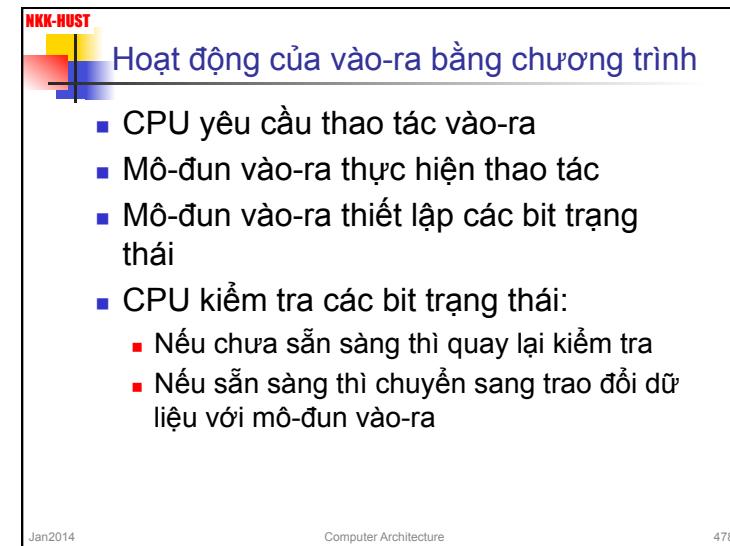
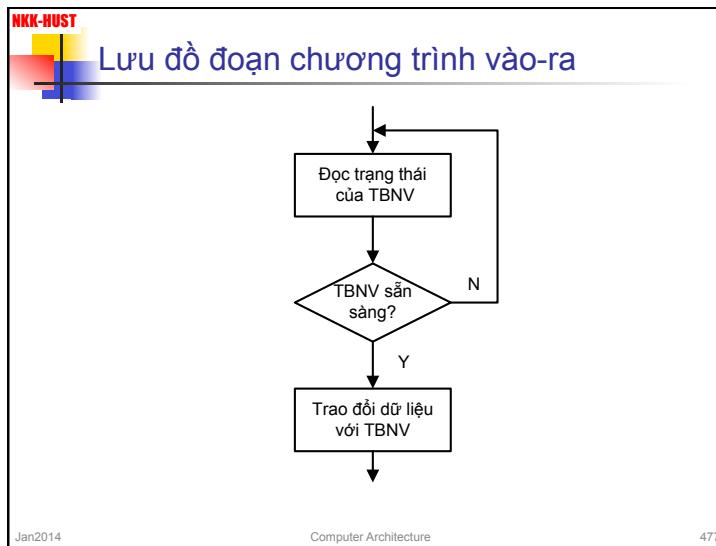
- Vào-ra bằng chương trình
(Programmed IO)
- Vào-ra điều khiển bằng ngắt
(Interrupt Driven IO)
- Truy nhập bộ nhớ trực tiếp - DMA
(Direct Memory Access)

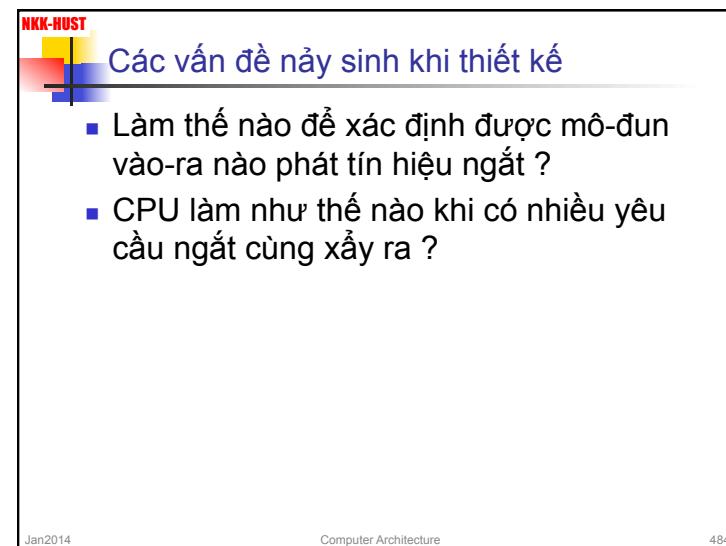
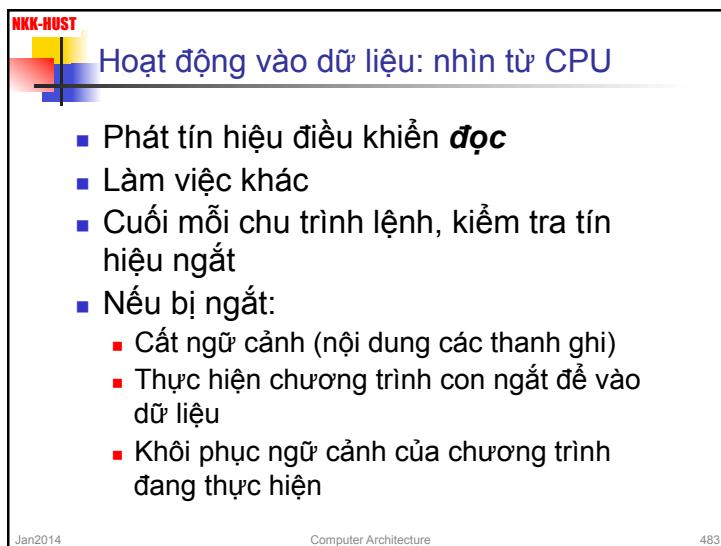
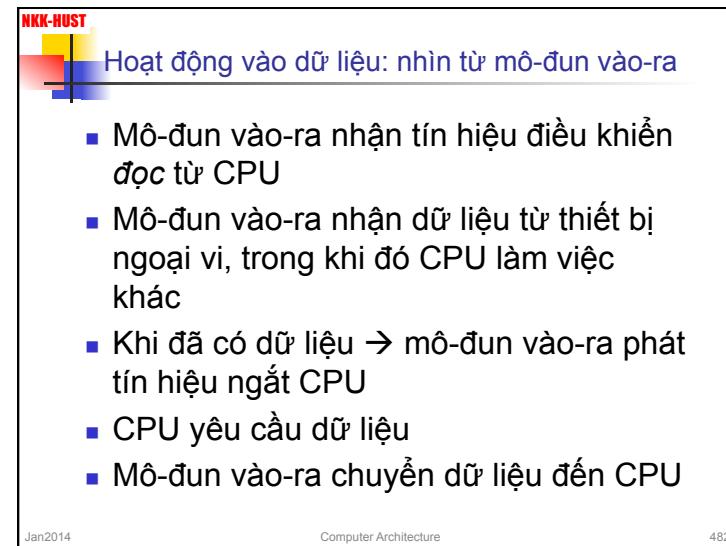
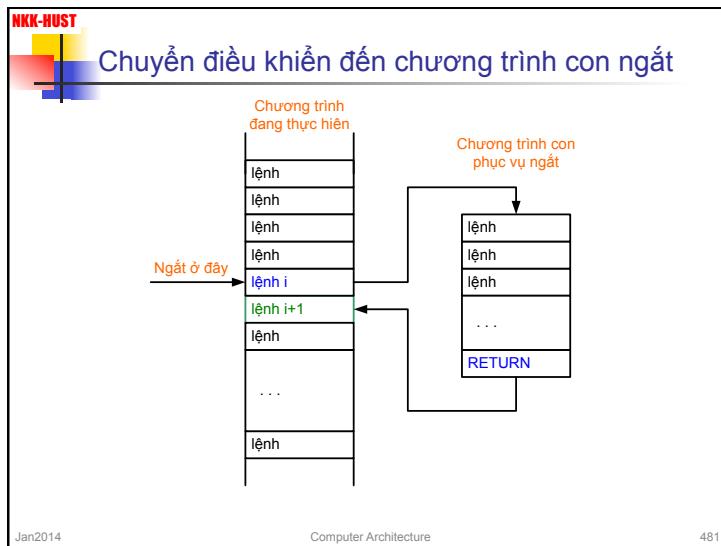
Jan2014

Computer Architecture

472







NKK-HUST

Các phương pháp nối ghép ngắn

- Sử dụng nhiều đường yêu cầu ngắn
- Hỏi vòng bằng phần mềm (Software Poll)
- Hỏi vòng bằng phần cứng (Daisy Chain or Hardware Poll)
- Sử dụng bộ điều khiển ngắn (PIC)

Jan2014 Computer Architecture 485

NKK-HUST

Nhiều đường yêu cầu ngắn

Mỗi mô-đun vào-ra được nối với một đường yêu cầu ngắn
CPU phải có nhiều đường tín hiệu yêu cầu ngắn
Hạn chế số lượng mô-đun vào-ra
Các đường ngắn được qui định mức ưu tiên

Jan2014 Computer Architecture 486

NKK-HUST

Hỏi vòng bằng phần mềm

- CPU thực hiện phần mềm hỏi lần lượt từng mô-đun vào-ra
- Chậm
- Thứ tự các mô-đun được hỏi vòng chính là thứ tự ưu tiên

Jan2014 Computer Architecture 487

NKK-HUST

Hỏi vòng bằng phần cứng

Jan2014 Computer Architecture 488

NKK-HUST

Kiểm tra vòng bằng phần cứng (tiếp)

- CPU phát tín hiệu chấp nhận ngắt (INTA) đến mô-đun vào-ra đầu tiên
- Nếu mô-đun vào-ra đó không gây ra ngắt thì nó gửi tín hiệu đến mô-đun kế tiếp cho đến khi xác định được mô-đun gây ngắt
- Thứ tự các mô-đun vào-ra kết nối trong chuỗi xác định thứ tự ưu tiên

Jan2014 Computer Architecture 489

NKK-HUST

Bộ điều khiển ngắt lập trình được

- PIC – Programmable Interrupt Controller
- PIC có nhiều đường vào yêu cầu ngắt có qui định mức ưu tiên
- PIC chọn một yêu cầu ngắt không bị cấm có mức ưu tiên cao nhất gửi tới CPU

Jan2014 Computer Architecture 490

NKK-HUST

Đặc điểm của vào-ra điều khiển bằng ngắt

- Có sự kết hợp giữa phần cứng và phần mềm
 - Phần cứng: gây ngắt CPU
 - Phần mềm: trao đổi dữ liệu
- CPU trực tiếp điều khiển vào-ra
- CPU không phải đợi mô-đun vào-ra → hiệu quả sử dụng CPU tốt hơn

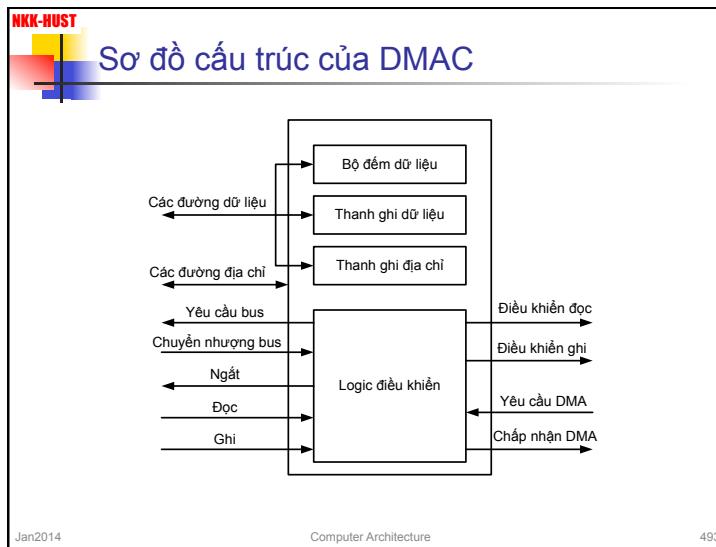
Jan2014 Computer Architecture 491

NKK-HUST

3. DMA (Direct Memory Access)

- Vào-ra bằng chương trình và bằng ngắt do CPU trực tiếp điều khiển:
 - Chiếm thời gian của CPU
 - Tốc độ truyền bị hạn chế vì phải chuyển qua CPU
- Để khắc phục dùng DMA
 - Thêm mô-đun phần cứng trên bus → DMAC (Controller)
 - DMAC điều khiển trao đổi dữ liệu giữa mô-đun vào-ra với bộ nhớ chính

Jan2014 Computer Architecture 492



- NKK-HUST**
- ### Các thành phần của DMAC
- Thanh ghi dữ liệu: chứa dữ liệu trao đổi
 - Thanh ghi địa chỉ: chứa địa chỉ ngăn nhớ dữ liệu
 - Bộ đếm dữ liệu: chứa số từ dữ liệu cần trao đổi
 - Logic điều khiển: điều khiển hoạt động của DMAC
- Jan2014 Computer Architecture 494

- NKK-HUST**
- ### Hoạt động DMA
- CPU “nói” cho DMAC
 - Vào hay Ra dữ liệu
 - Địa chỉ thiết bị vào-ra (cổng vào-ra tương ứng)
 - Địa chỉ đầu của mảng nhớ chứa dữ liệu → nạp vào thanh ghi địa chỉ
 - Số từ dữ liệu cần truyền → nạp vào bộ đếm dữ liệu
 - CPU làm việc khác
 - DMA điều khiển trao đổi dữ liệu
 - Sau khi truyền được một từ dữ liệu thì:
 - nội dung thanh ghi địa chỉ tăng
 - nội dung bộ đếm dữ liệu giảm
 - Khi bộ đếm dữ liệu = 0, DMAC gửi tín hiệu ngắt CPU để báo kết thúc DMA
- Jan2014 Computer Architecture 495

- NKK-HUST**
- ### Các kiểu thực hiện DMA
- DMA truyền theo khối (Block-transfer DMA): DMAC sử dụng bus để truyền xong cả khối dữ liệu
 - DMA lấy chu kỳ (Cycle Stealing DMA): DMAC cưỡng bức CPU treo tạm thời từng chu kỳ bus, DMAC chiếm bus thực hiện truyền một từ dữ liệu.
 - DMA trong suốt (Transparent DMA): DMAC nhận biết những chu kỳ nào CPU không sử dụng bus thì chiếm bus để trao đổi một từ dữ liệu.
- Jan2014 Computer Architecture 496

Cấu hình DMA (1)

```

graph TD
    SB[System Bus] --- CPU[CPU]
    SB --- DMAC[DMAC]
    SB --- IM1[I/O Module]
    SB --- IM2[I/O Module]
    SB --- IM3[I/O Module]
    SB --- Mem[Memory]
    DMAC --- IM1
    DMAC --- IM2
  
```

- Mỗi lần trao đổi một dữ liệu, DMAC sử dụng bus hai lần
 - Giữa mô-đun vào-ra với DMAC
 - Giữa DMAC với bộ nhớ

Jan2014 Computer Architecture 497

Cấu hình DMA (2)

```

graph TD
    SB[System Bus] --- CPU[CPU]
    SB --- DMAC[DMAC]
    SB --- Mem[Memory]
    DMAC --- IM1[I/O Module]
    DMAC --- IM2[I/O Module]
  
```

- DMAC điều khiển một hoặc vài mô-đun vào-ra
- Mỗi lần trao đổi một dữ liệu, DMAC sử dụng bus một lần
 - Giữa DMAC với bộ nhớ

Jan2014 Computer Architecture 498

Cấu hình DMA (3)

```

graph TD
    SB[System Bus] --- CPU[CPU]
    SB --- DMAC[DMAC]
    SB --- Mem[Memory]
    DMAC --- IOB[IO Bus]
    IOB --- IM1[I/O Module]
    IOB --- IM2[I/O Module]
    IOB --- IM3[I/O Module]
  
```

- Bus vào-ra tách rời hỗ trợ tất cả các thiết bị cho phép DMA
- Mỗi lần trao đổi một dữ liệu, DMAC sử dụng bus một lần
 - Giữa DMAC với bộ nhớ

Jan2014 Computer Architecture 499

Đặc điểm của DMA

- CPU không tham gia trong quá trình trao đổi dữ liệu
- DMAC điều khiển trao đổi dữ liệu giữa bộ nhớ chính với mô-đun vào-ra (hoàn toàn bằng phần cứng) → tốc độ nhanh
- Phù hợp với các yêu cầu trao đổi mảng dữ liệu có kích thước lớn

Jan2014 Computer Architecture 500

4. Bộ xử lý vào-ra

- Việc điều khiển vào-ra được thực hiện bởi một bộ xử lý vào-ra chuyên dụng
- Bộ xử lý vào-ra hoạt động theo chương trình của riêng nó
- Chương trình của bộ xử lý vào-ra có thể nằm trong bộ nhớ chính hoặc nằm trong một bộ nhớ riêng
- Hoạt động theo kiến trúc đa xử lý

Jan2014

Computer Architecture

501

8.3. Nối ghép thiết bị ngoại vi

1. Các kiểu nối ghép vào-ra

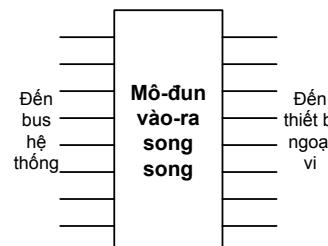
- Nối ghép song song
- Nối ghép nối tiếp

Jan2014

Computer Architecture

502

Nối ghép song song



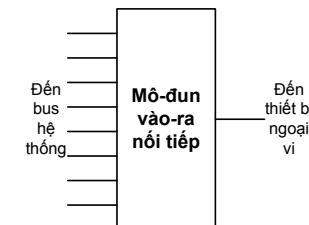
- Truyền nhiều bit song song
- Tốc độ nhanh
- Cần nhiều đường truyền dữ liệu

Jan2014

Computer Architecture

503

Nối ghép nối tiếp



- Truyền lần lượt từng bit
- Cần có bộ chuyển đổi từ dữ liệu song song sang nối tiếp hoặc/và ngược lại
- Tốc độ chậm hơn
- Cần ít đường truyền dữ liệu

Jan2014

Computer Architecture

504

2. Các cấu hình nối ghép

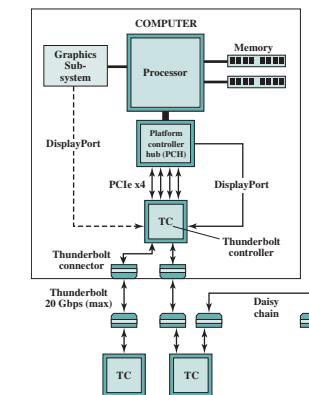
- Điểm tới điểm (Point to Point)
 - Thông qua một cổng vào-ra nối ghép với một thiết bị ngoại vi
- Điểm tới đa điểm (Point to Multipoint)
 - Thông qua một cổng vào-ra cho phép nối ghép được với nhiều thiết bị ngoại vi
 - Ví dụ:
 - USB (Universal Serial Bus): 127 thiết bị
 - IEEE 1394 (FireWire): 63 thiết bị
 - Thunderbolt

Jan2014

Computer Architecture

505

Thunderbolt



Jan2014

Computer Architecture

506

Hết chương 8

Jan2014

Computer Architecture

507

Kiến trúc máy tính

Chương 9 CÁC KIẾN TRÚC SONG SONG

Nguyễn Kim Khánh
Trường Đại học Bách khoa Hà Nội

Jan2014

Computer Architecture

508

Nội dung học phần

- Chương 1. Giới thiệu chung
- Chương 2. Cơ bản về logic số
- Chương 3. Hệ thống máy tính
- Chương 4. Số học máy tính
- Chương 5. Kiến trúc tập lệnh
- Chương 6. Bộ xử lý
- Chương 7. Bộ nhớ máy tính
- Chương 8. Hệ thống vào-ra
- Chương 9. Các kiến trúc song song**

Jan2014 Computer Architecture 509

Nội dung của chương 9

- 9.1. Phân loại kiến trúc máy tính
- 9.2. Hệ thống đa xử lý bộ nhớ dùng chung
- 9.3. Hệ thống đa xử lý bộ nhớ phân tán
- 9.4. GPGPU

Jan2014 Computer Architecture 510

9.1. Phân loại kiến trúc máy tính

Phân loại kiến trúc máy tính (Michael Flynn -1966)

- SISD - Single Instruction Stream, Single Data Stream
- SIMD - Single Instruction Stream, Multiple Data Stream
- MISD - Multiple Instruction Stream, Single Data Stream
- MIMD - Multiple Instruction Stream, Multiple Data Stream

Jan2014 Computer Architecture 511

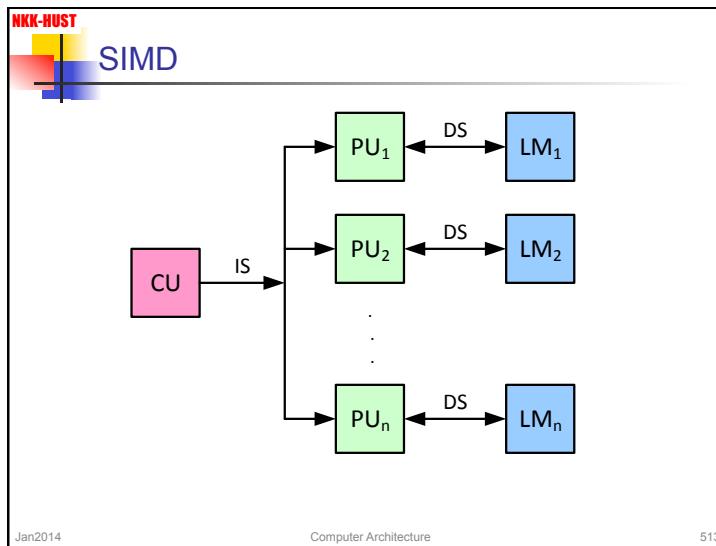
SISD

```

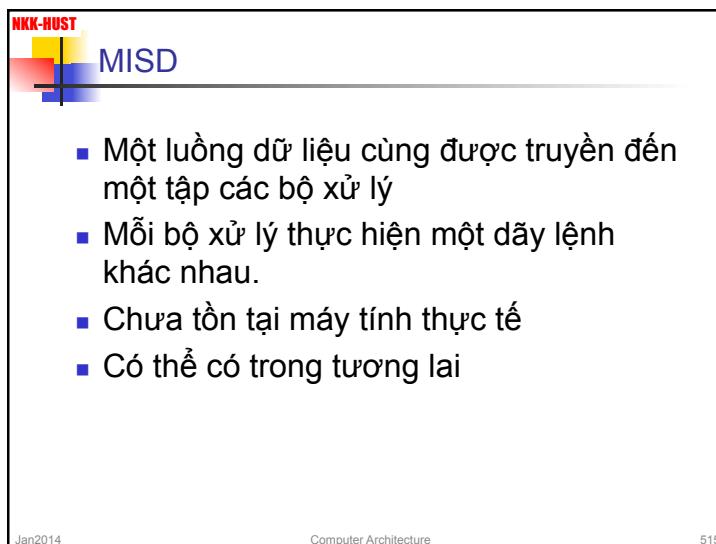
graph LR
    CU[CU] -- IS --> PU[PU]
    PU -- DS --> MU[MU]
  
```

- CU: Control Unit
- PU: Processing Unit
- MU: Memory Unit
- Một bộ xử lý
- Đơn dòng lệnh
- Dữ liệu được lưu trữ trong một bộ nhớ
- Chính là Kiến trúc von Neumann (tuần tự)

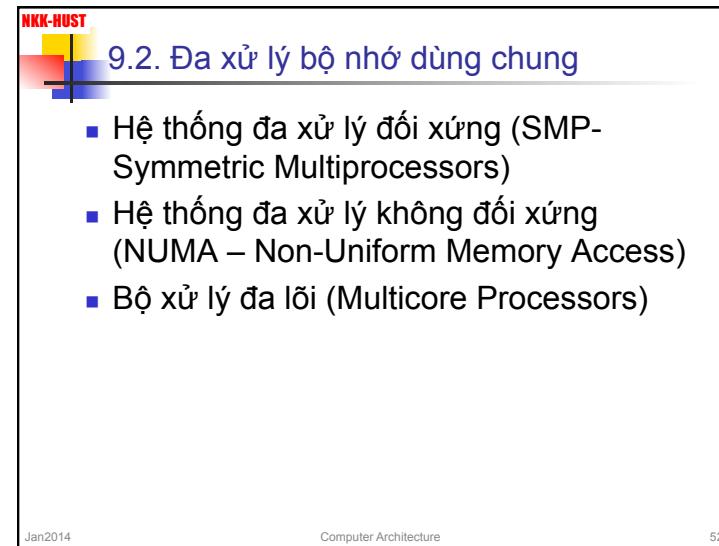
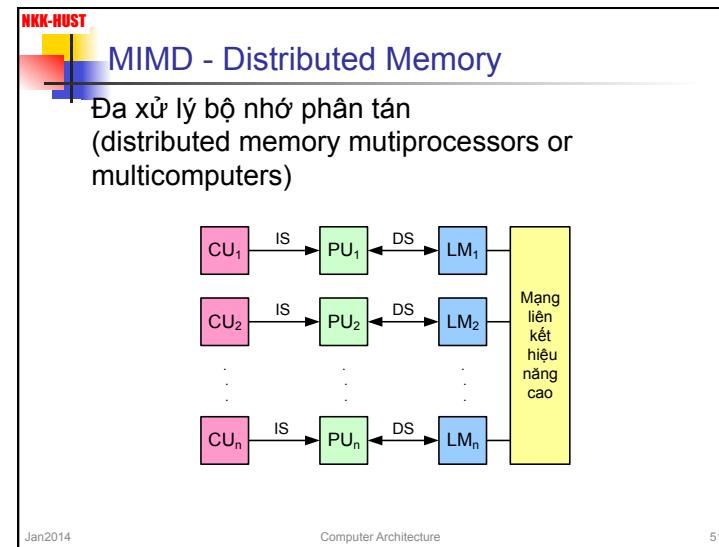
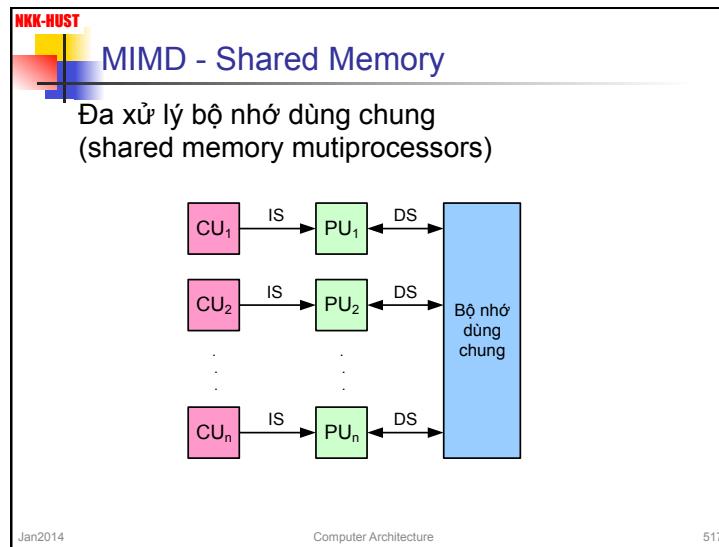
Jan2014 Computer Architecture 512

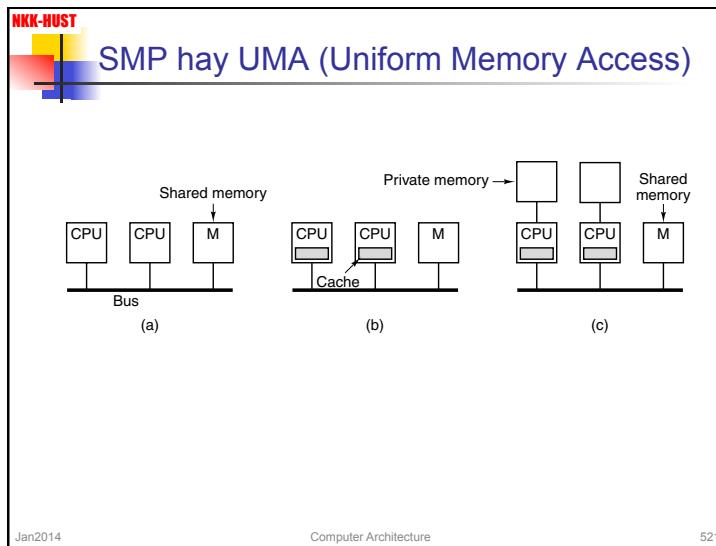


- NKK-HUST**
- ### SIMD (tiếp)
- Đơn dòng lệnh điều khiển đồng thời các đơn vị xử lý PUs
 - Mỗi phần tử xử lý có một bộ nhớ dữ liệu riêng LM (local memory)
 - Mỗi lệnh được thực hiện trên một tập các dữ liệu khác nhau
 - Các mô hình SIMD
 - Vector Computer
 - Array processor
- Jan2014 Computer Architecture 514

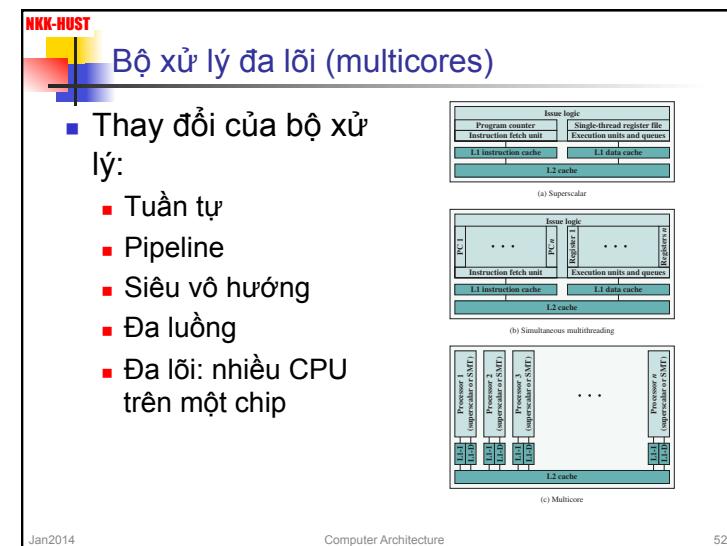
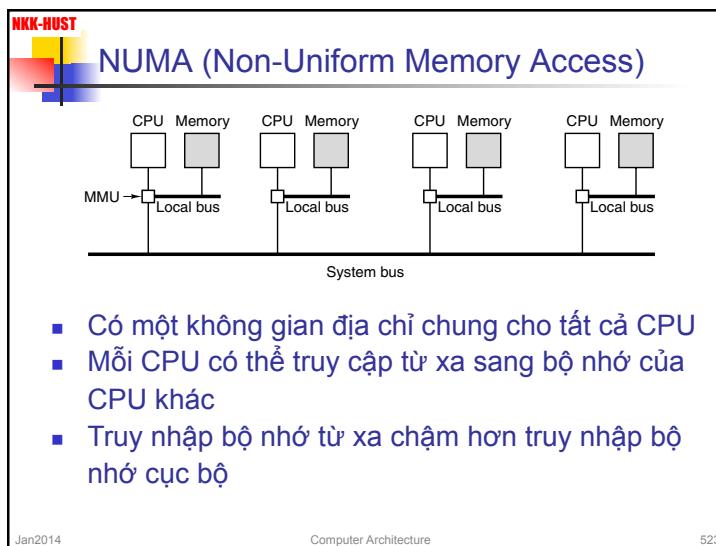


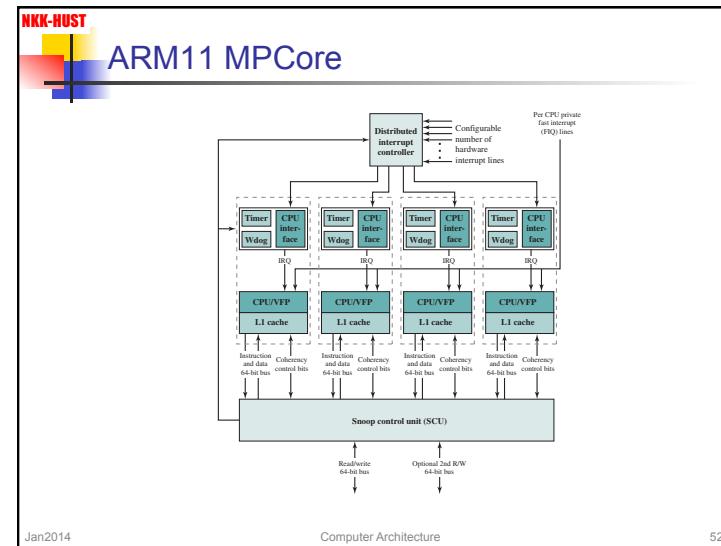
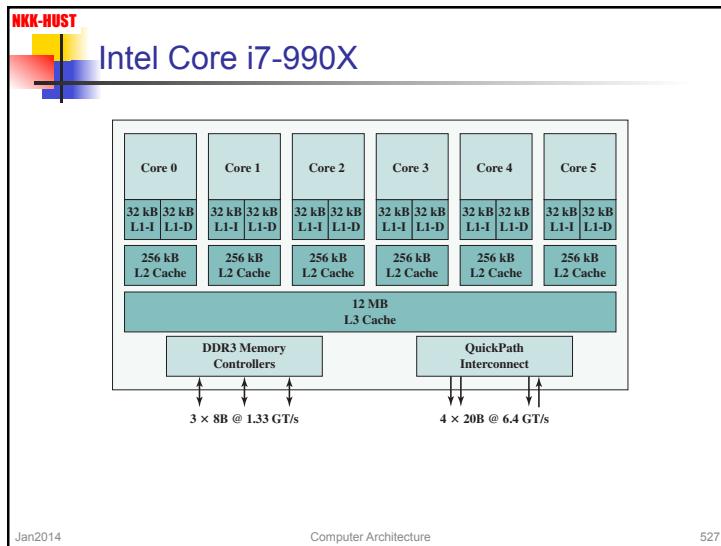
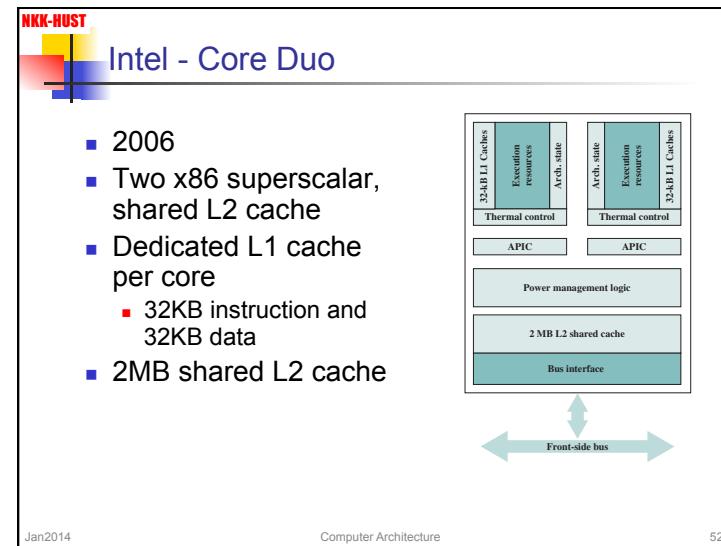
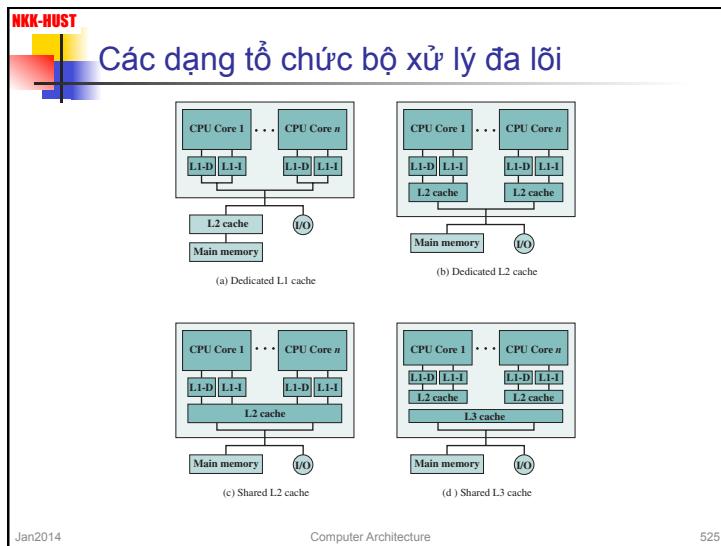
- NKK-HUST**
- ### MIMD
- Tập các bộ xử lý
 - Các bộ xử lý đồng thời thực hiện các dãy lệnh khác nhau trên các dữ liệu khác nhau
 - Các mô hình MIMD
 - Multiprocessors (Shared Memory)
 - Multicomputers (Distributed Memory)
- Jan2014 Computer Architecture 516





- NKK-HUST**
- ### SMP (tiếp)
- Một máy tính có $n \geq 2$ bộ xử lý giống nhau
 - Các bộ xử lý dùng chung bộ nhớ và hệ thống vào-ra
 - Thời gian truy cập bộ nhớ là bằng nhau với các bộ xử lý
 - Các bộ xử lý có thể thực hiện chức năng giống nhau
 - Hệ thống được điều khiển bởi một hệ điều hành phân tán
 - Hiệu năng: Các công việc có thể thực hiện song song
 - Khả năng chịu lỗi
- Jan2014 Computer Architecture 522





NKK-HUST

9.3. Đa xử lý bộ nhớ phân tán

Máy tính qui mô lớn (Warehouse Scale Computers or Massively Parallel Processors – MPP)

Máy tính cụm (clusters)

Jan2014 Computer Architecture 529

NKK-HUST

Mạng liên kết

Jan2014 Computer Architecture 530

NKK-HUST

Massively Parallel Processors

- Hệ thống qui mô lớn
- Đắt tiền: nhiều triệu USD
- Dùng cho tính toán khoa học và các bài toán có số phép toán và dữ liệu rất lớn
- Siêu máy tính

Jan2014 Computer Architecture 531

NKK-HUST

IBM Blue Gene/P

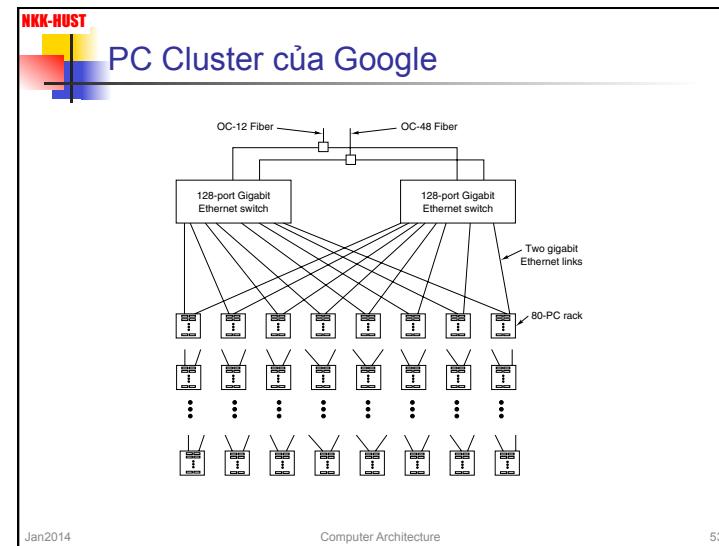
Chip:	4 processors 8-MB L3 cache	Card:	1 Chip 4 CPUs 2 GB	Board:	32 Cards 32 Chips 128 CPUs 64 GB	Cabinet:	32 Boards 1024 Cards 4096 CPUs 2 TB	System:	72 Cabinets 73728 Cards 73728 Chips 294912 CPUs 144 TB
(a)	(b)	(c)	(d)	(e)					

Jan2014 Computer Architecture 532

Cluster

- Nhiều máy tính được kết nối với nhau bằng mạng liên kết tốc độ cao (~ Gbps)
- Mỗi máy tính có thể làm việc độc lập (PC hoặc SMP)
- Mỗi máy tính được gọi là một node
- Các máy tính có thể được quản lý làm việc song song theo nhóm (cluster)
- Toàn bộ hệ thống có thể coi như là một máy tính song song
- Tính sẵn sàng cao
- Khả năng chịu lỗi lớn

Jan2014 Computer Architecture 533



9.4. Bộ xử lý đồ họa tính toán đa năng

- Kiến trúc SIMD
- Xuất phát từ bộ xử lý đồ họa GPU (Graphic Processing Unit) hỗ trợ xử lý đồ họa 2D và 3D: xử lý dữ liệu song song
- GPGPU – General purpose Graphic Processing Unit
- Hệ thống lai CPU/GPGPU
 - CPU là host: thực hiện theo tuần tự
 - GPGPU: tính toán song song

Jan2014 Computer Architecture 535

