Kiến trúc máy tính
Computer architecture

To improve is to change; to be perfect is to change often.
Winston Churchill

Nội dung môn học

- Chương 1: Giới thiệu
- Chương 2: Kiến trúc tập lệnh MIPS – 32 bits
- Chương 3: Bộ xử lý MIPS – 32 bits
- Chương 4: Xử lý đường ống
- Chương 5: Bộ nhớ máy tính
- Chương 6: Tổ chức Vào/ra

Tài liệu tham khảo

- http://www-inst.eecs.berkeley.edu/~cs61c/fa16/ và
- http://inst.eecs.berkeley.edu/~cs61c/sp15/
- Computer Organization and Design, 5th Edition : The Hardware/Software Interface

Lịch sử phát triển của máy tính

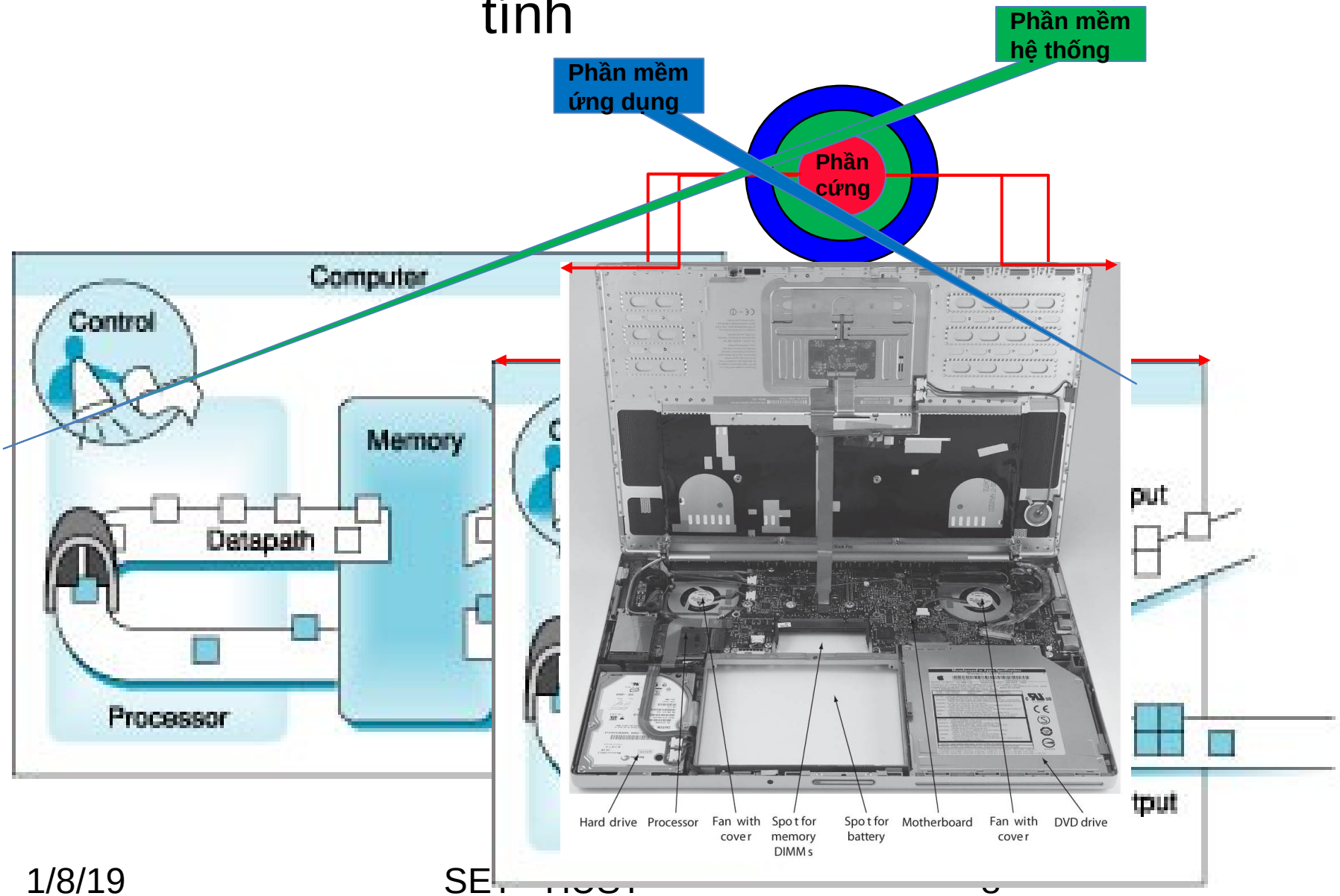
Máy tính hiện đại

Personal Mobile Devices

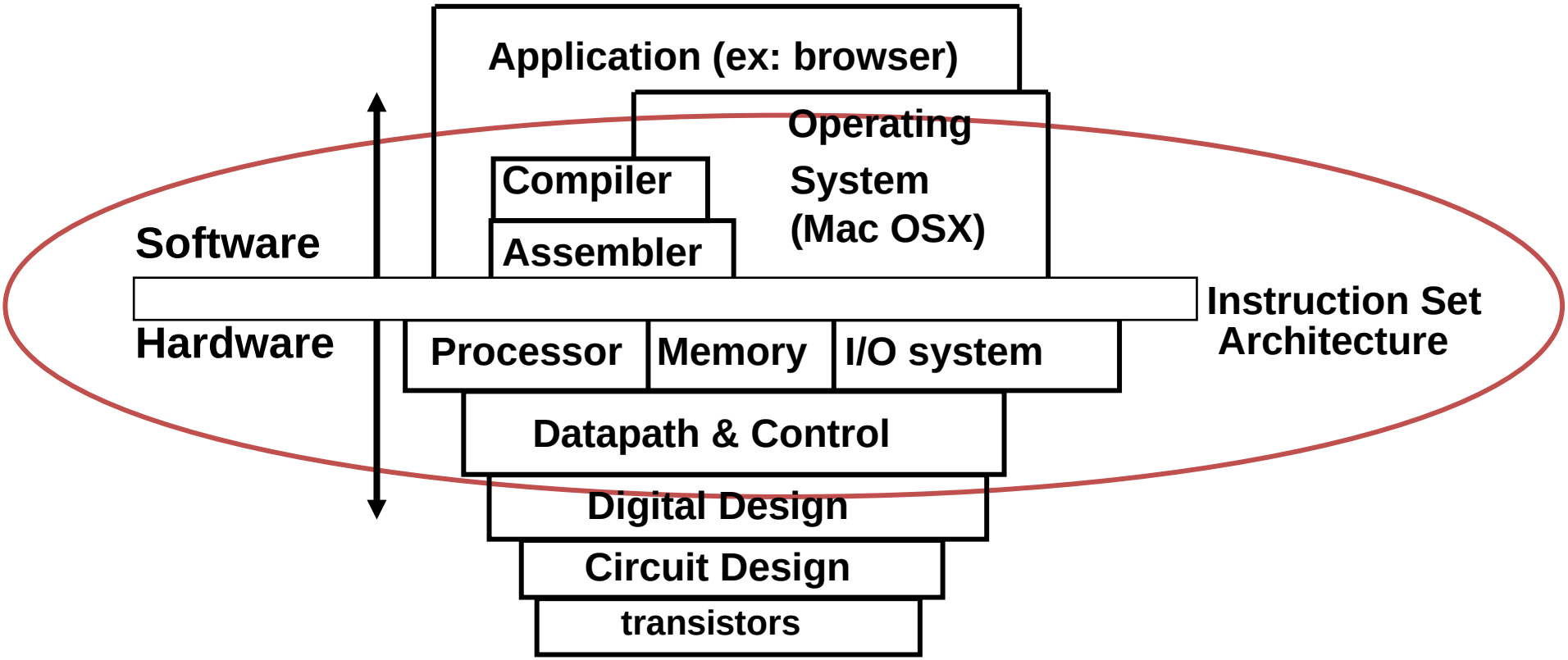
Máy tính hiện đại

Máy tính hiện đại

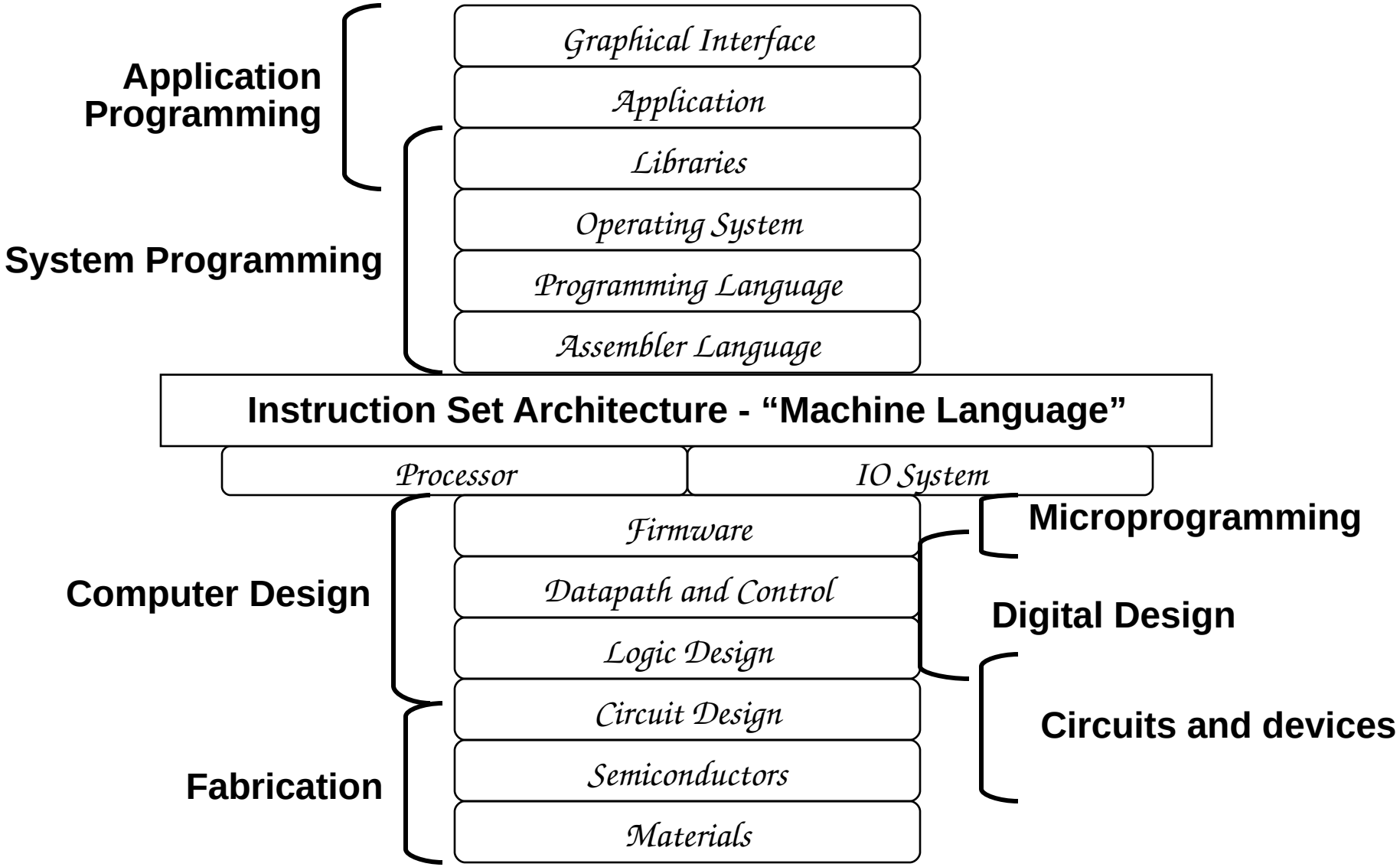
Quan niệm truyền thống về cấu trúc máy tính



Quan niệm truyền thống về cấu trúc máy tính

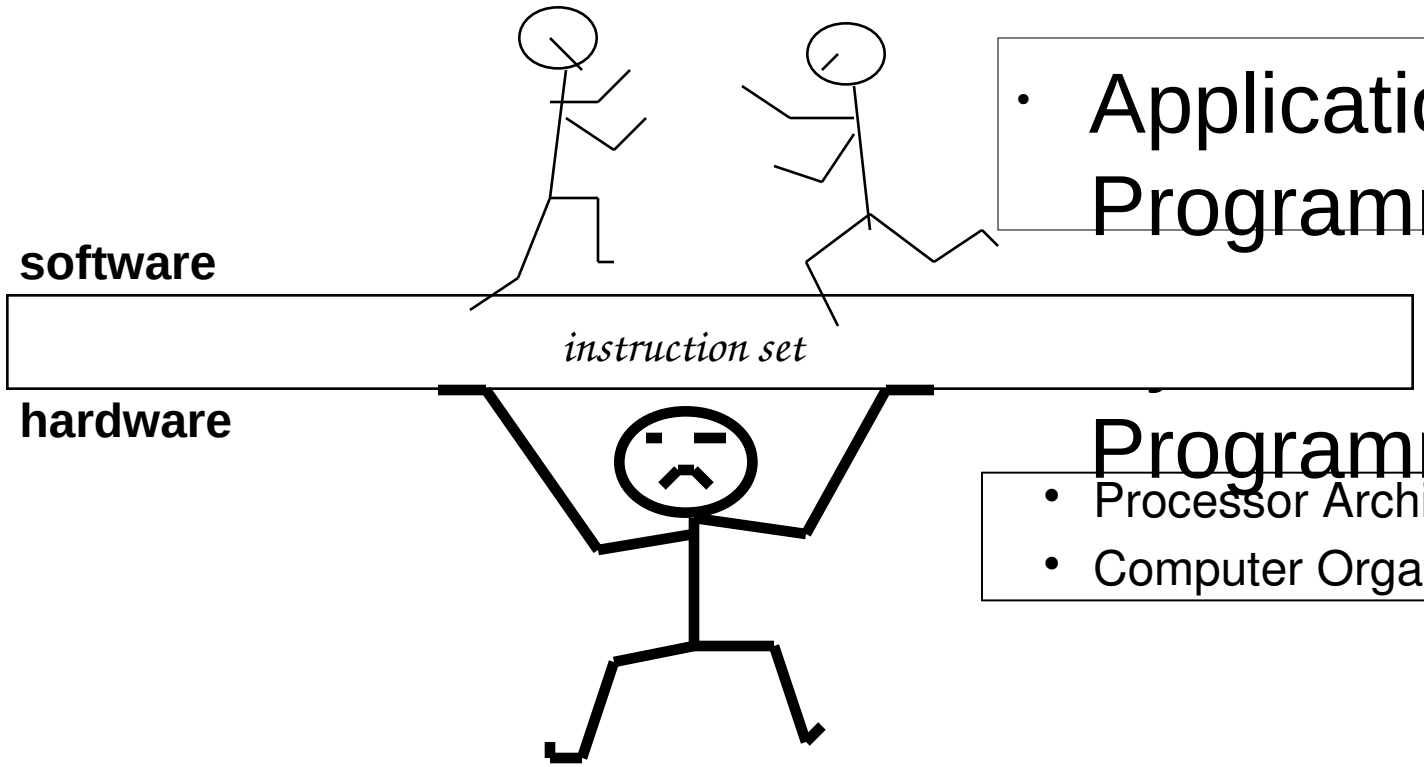


Cấu trúc máy tính



Kiến trúc máy tính?

Computer Architecture = Instruction Set Architecture + Machine Organization



- Application Programming

Programming

- Processor Architecture
- Computer Organization

Bắt đầu từ : **Nguyên lý thiết kế và cấu trúc máy tính truyền thống**

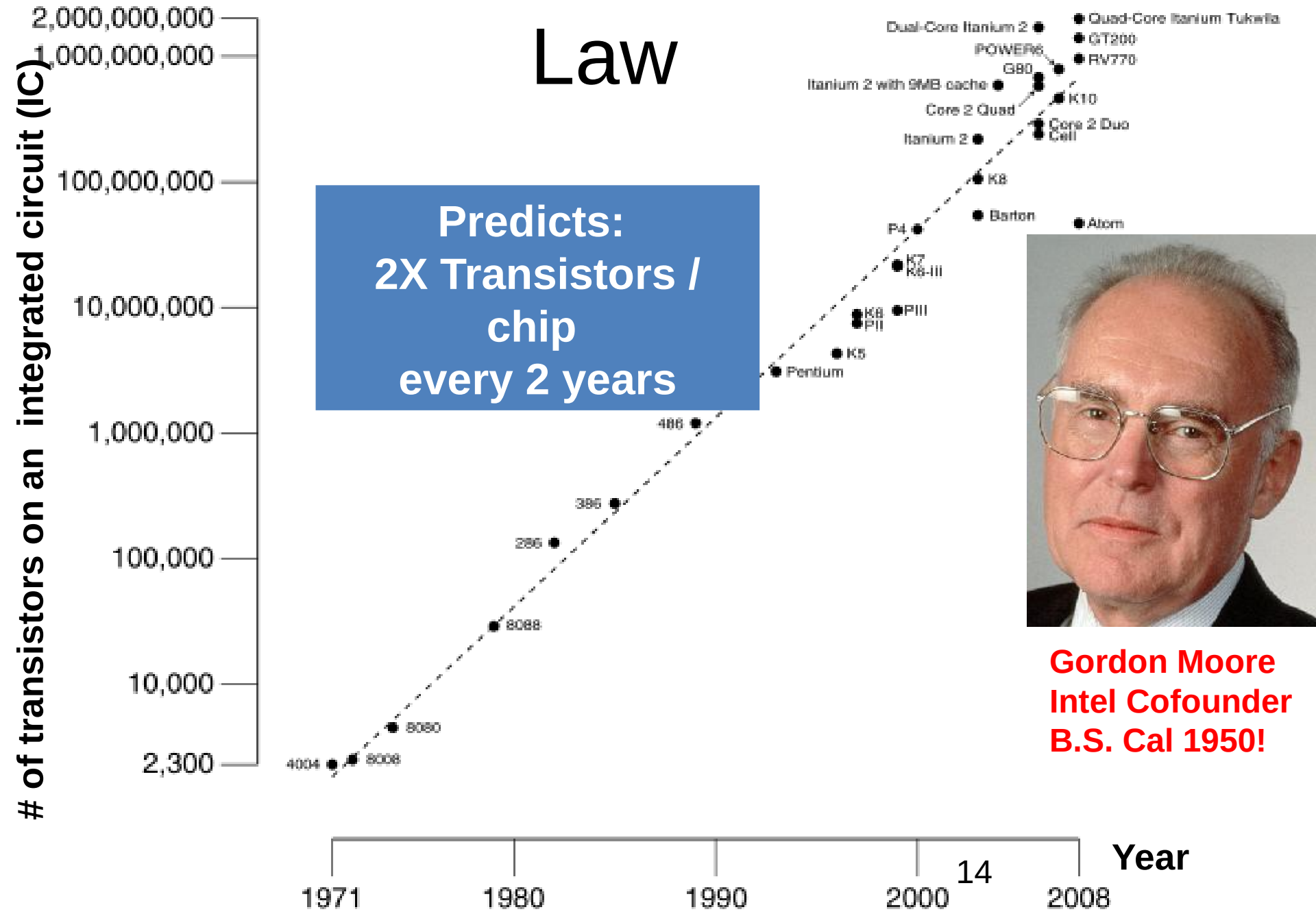
Hiểu nguyên lý thiết kế

Thiết kế máy tính theo yêu cầu

Cấu trúc bộ xử lý MIPS – 32
MIPS = Microprocessor without Interlocked Pipeline Stages

To change

Định luật Moore: Moore's Law



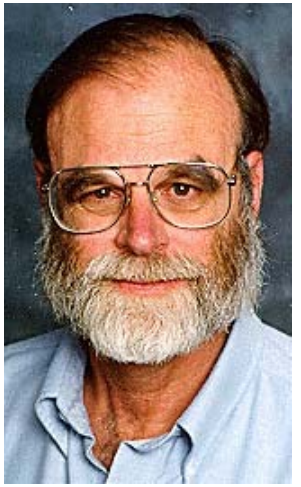
Gordon Moore
Intel Cofounder
B.S. Cal 1950!

Định luật Moore

Định luật Moore là một bước ngoặt lớn trong ngành công nghệ điện tử, giải thích tại sao nhà sản xuất có thể giảm giá thành trong khi vẫn tiếp tục nâng cao hiệu suất của phần cứng.

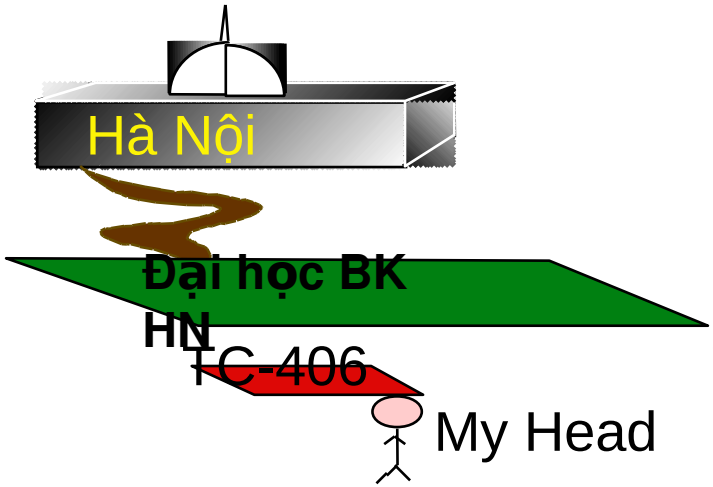
Hiện nay, thời gian để tăng đôi số transistor/inch vuông đã dài hơn vì kích thước transistor không thể giảm nhỏ kích thước phân tử <14nm (Hạn chế về thiết kế vật

Đánh giá khả năng lưu trữ trên bộ nhớ tương tự của Jim Gray
How Far Away is the Data?

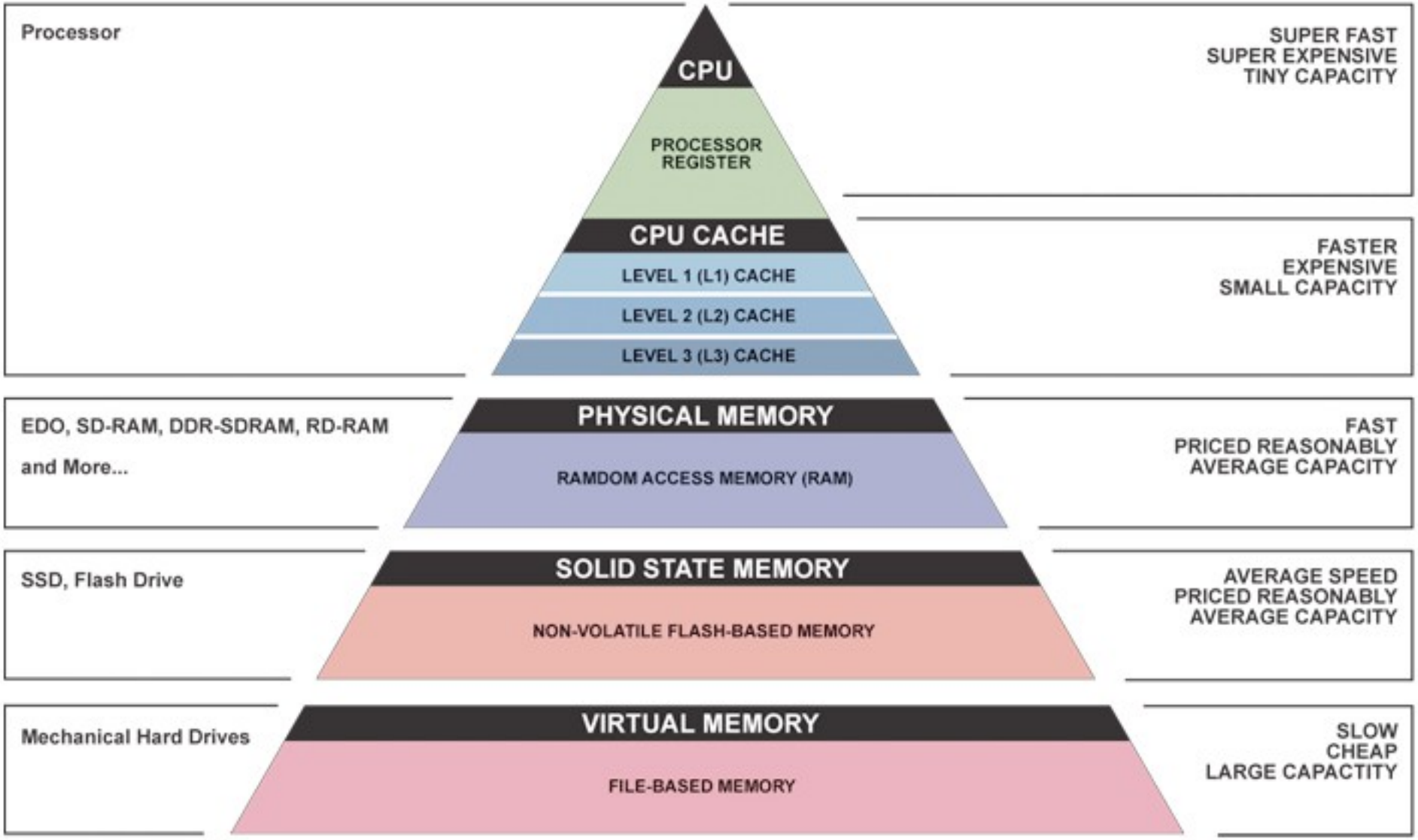


Jim Gray
Turing Award
B.S. Cal 1966
Ph.D. Cal 1969!

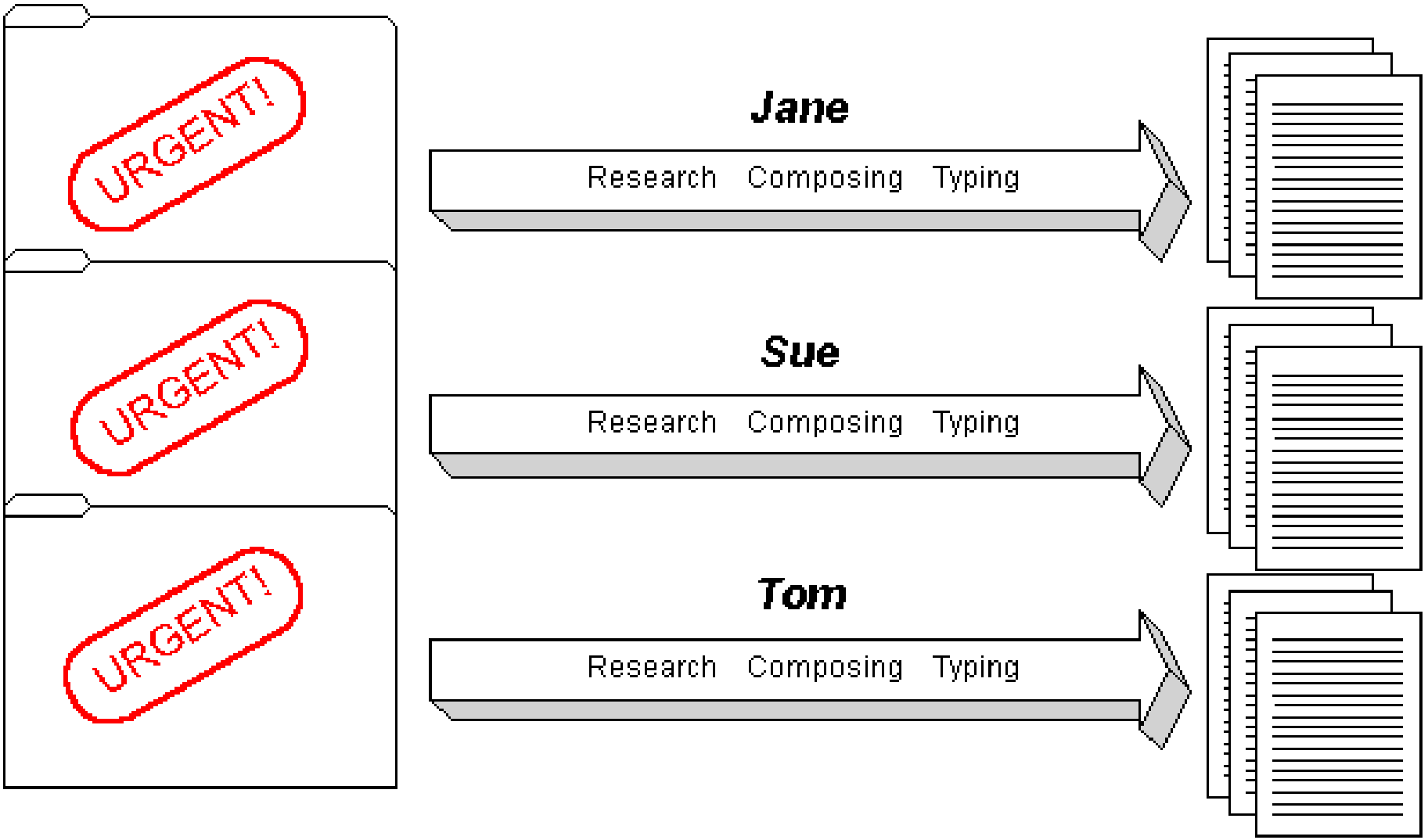
Dung lượng		Thời gian lưu trữ /truy cập
10^9	Tape /Optical	2,000 Years
10^0	Robot	
10^6	Disk	2 Years
10^0		
10^0	Memory	1.5 hr
10^1	On Board Cache	10 min
10^2	On Chip Cache	1 min
10^1	Registers	1 (ns)



Nguyên lý về định vị và phân cấp bô nhớ



Xử lý song song



Quan niệm mới (phức tạp hơn một chút!)

Software | *Hardware*

- Parallel Requests

Assigned to computer

e.g., Search “Katz”

- Parallel Thread

Assigned to core

e.g., Lookup, Ads

- Parallel Instructions

>1 instruction @ one time

e.g., 5 pipelined instructions

- Parallel Data

>1 data item @ one time

e.g., Add of 4 pairs of words

Harness Parallelism & Achieve High Performance

Warehous e-Scale Computer

Smart Phone

Computer

Core … Core

Memory (Cache)

Input/Output

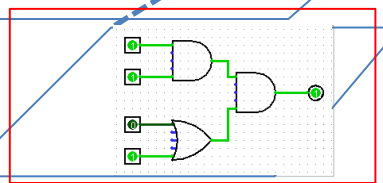
Cor e

Instruction Unit(s)

Fun ctional Unit(s)

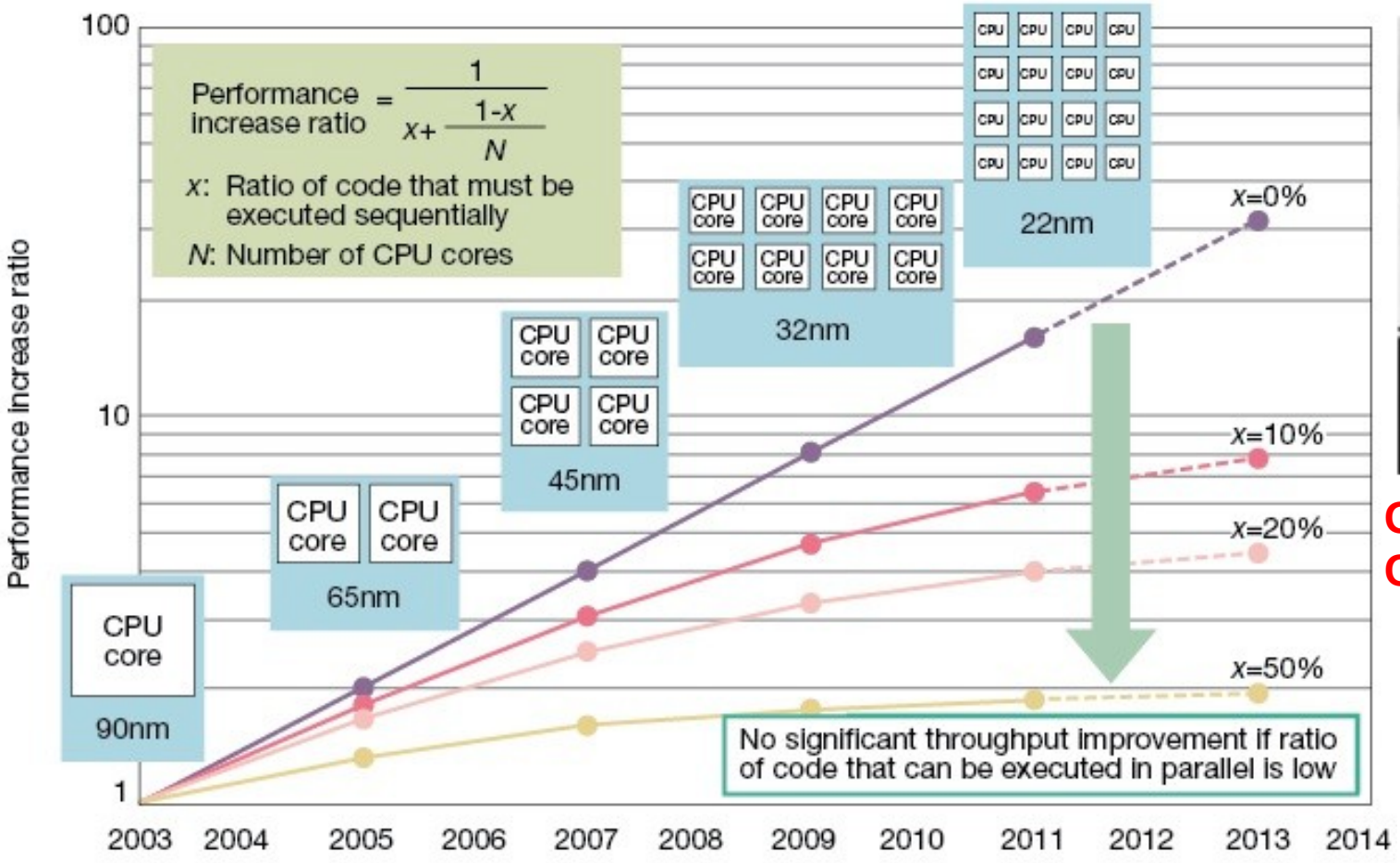
A0+B 0 | A1+B 1 | A2+B 2 | A3+B 3

Main Memory



Logic Gates

Luật Amdahl: Amdahl's Law

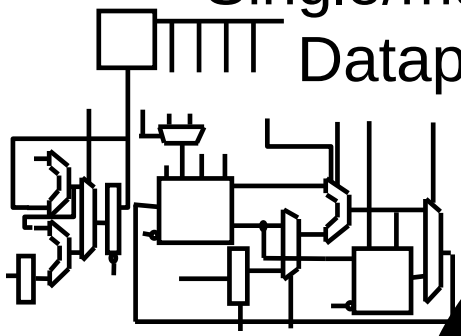


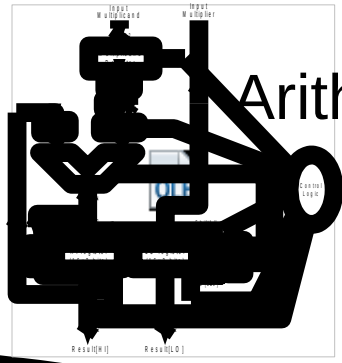
Gene Amdahl
Computer Pioneer

Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor performance. Diagram assumes no overhead in parallel processing. Years shown for design rules based on Intel planned and actual technology. Core count assumed to double for each rule generation.

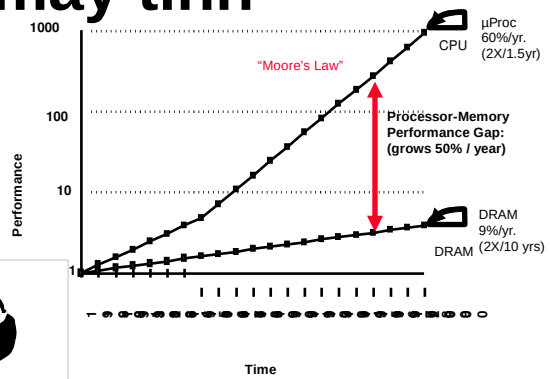
Tiến trình tìm hiểu về kiến trúc máy tính

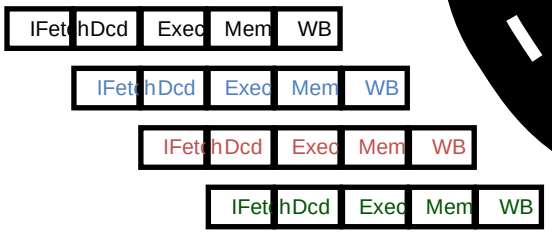
Single/multicycle Datapaths



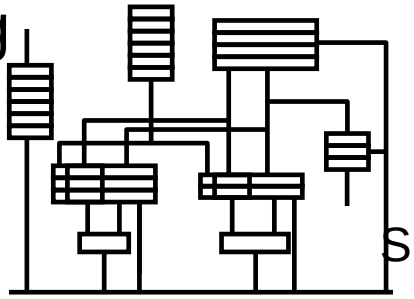


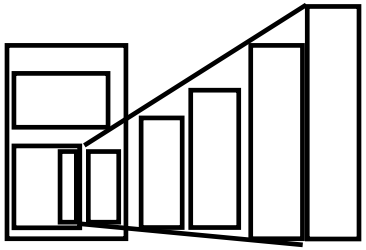
Arithmetic



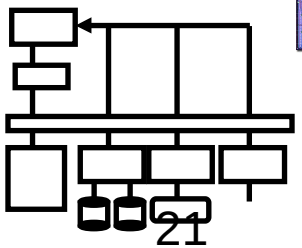


Pipelining

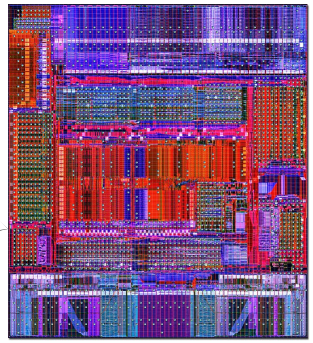




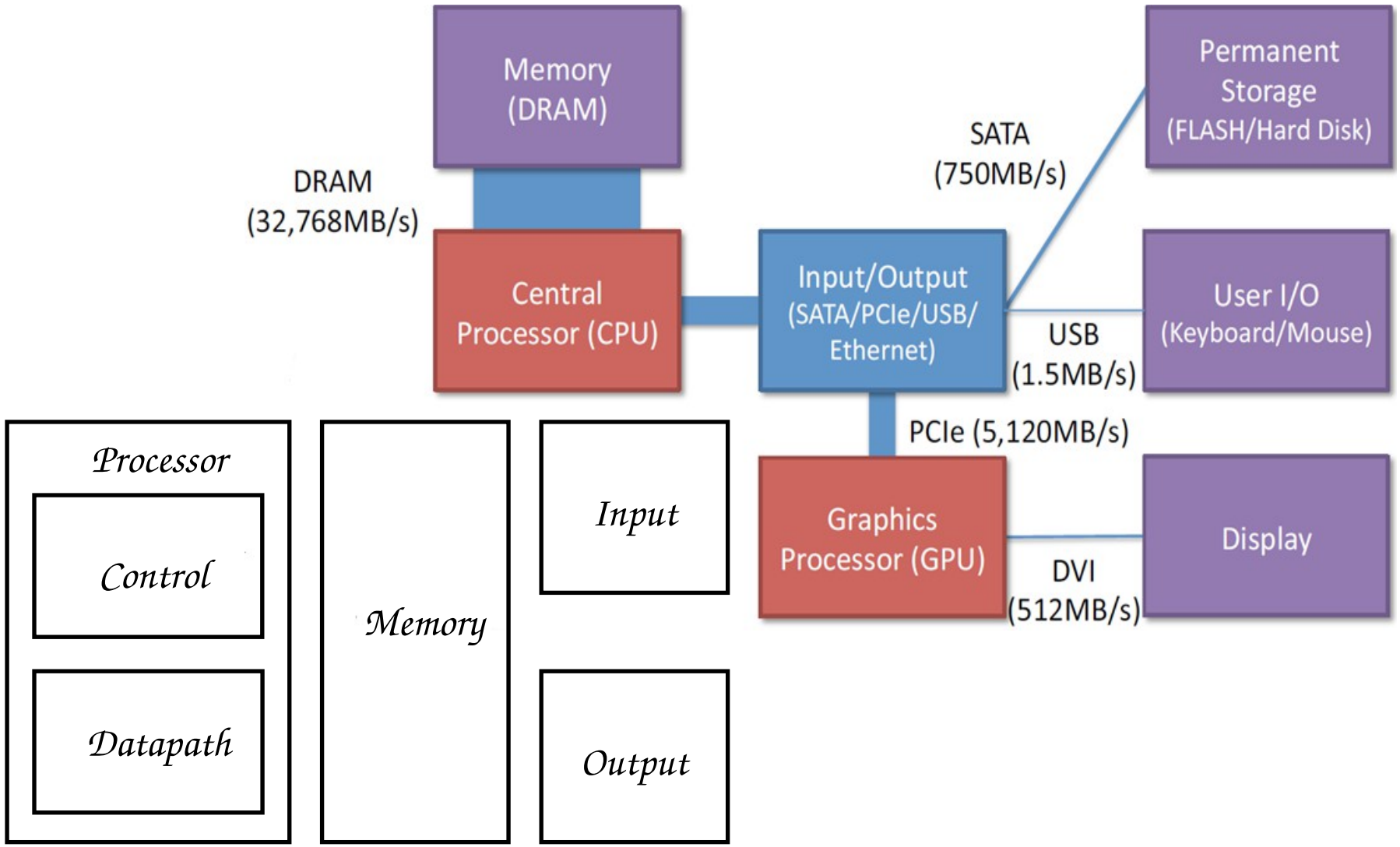
Memory Systems



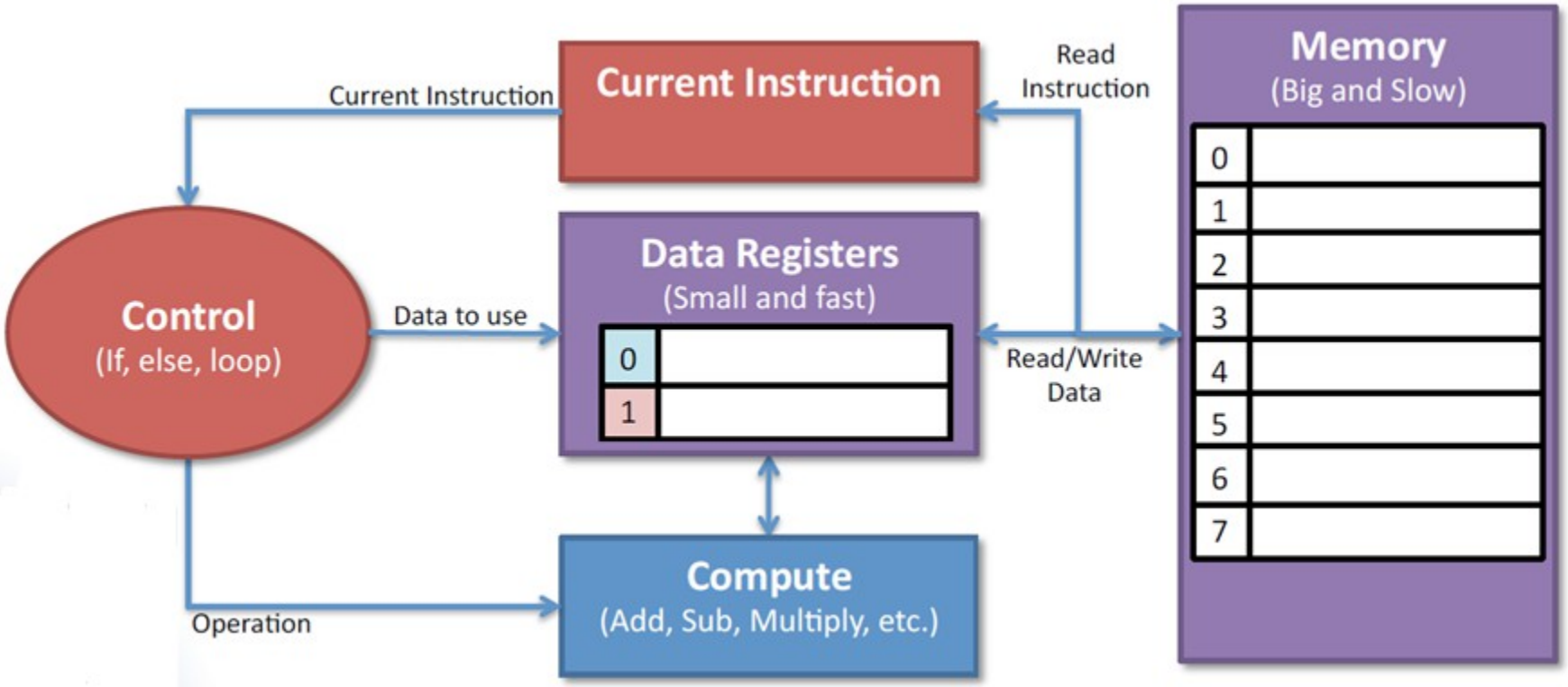
I/O



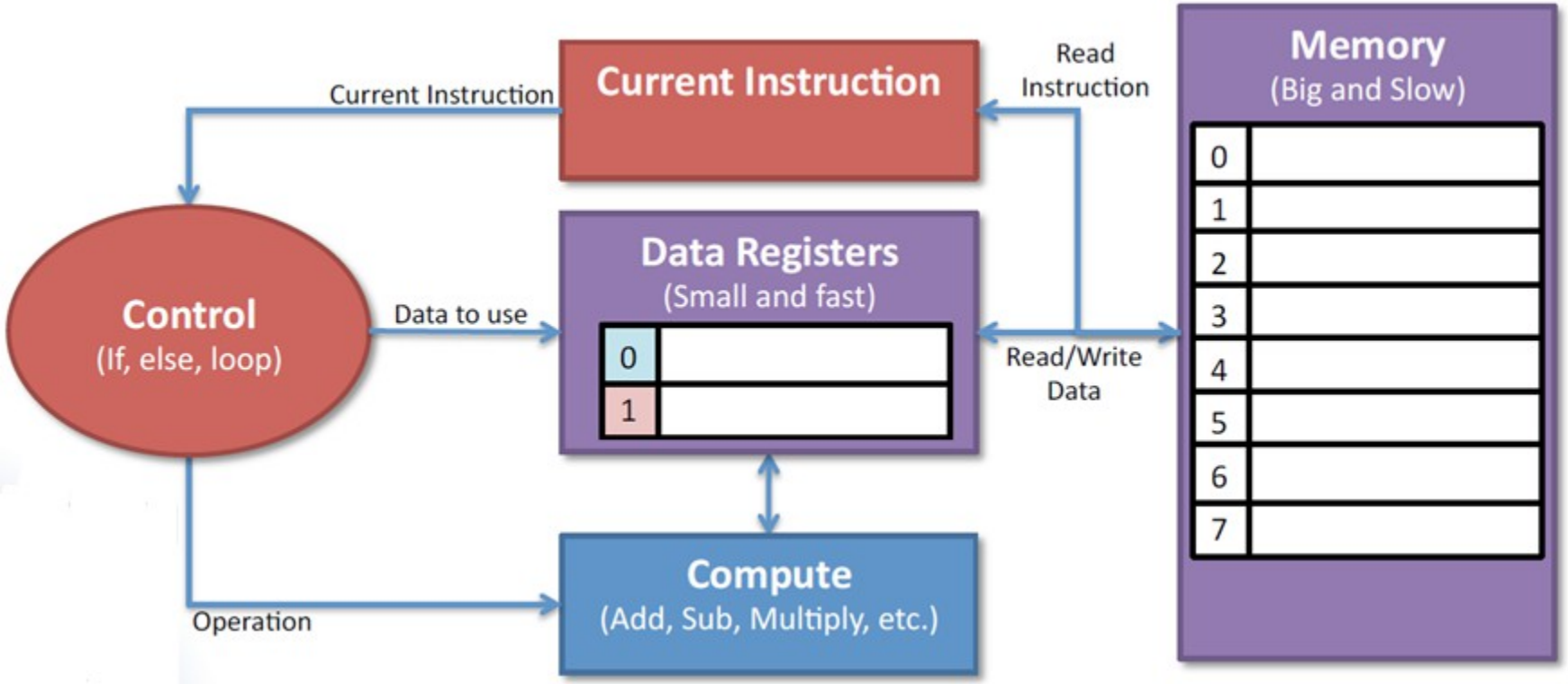
Cấu tạo của máy tính



Cấu tạo bộ xử lý



Bộ xử lý cơ bản: Bộ nhớ, Khối điều khiển,



Các cấp độ diễn tả trừu tượng

High Level Language Program (e.g., C)

Compiler

Assembly Language Program (e.g., MIPS)

Assembler

Machine Language Program (MIPS)

Machine Interpretation

Hardware Architecture Description (e.g., block diagrams)

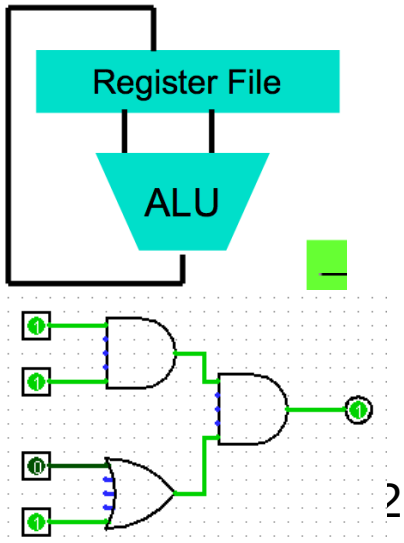
Architecture Implementation

Logic Circuit Description (Circuit Schematic Diagrams)

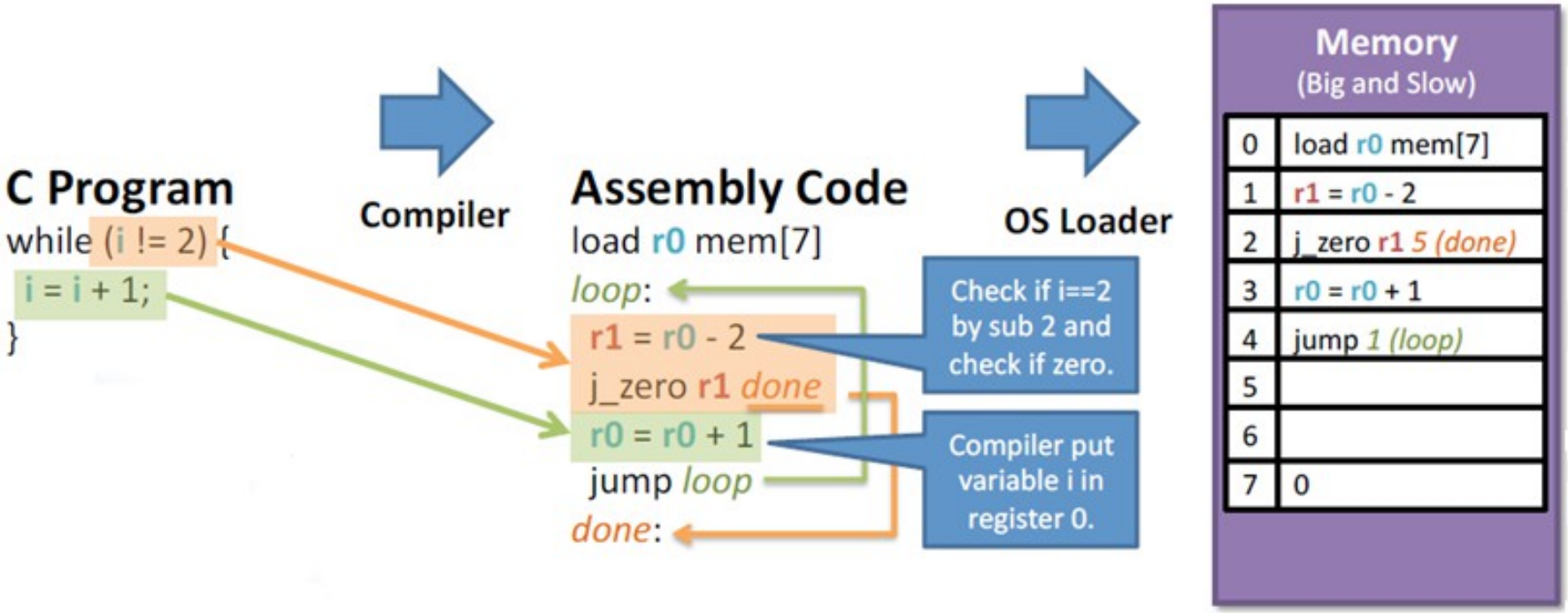
```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;

lw      $t0, 0($2)
lw      $t1, 4($2)
sw      $t1, 0($2)
sw      $t0, 4($2)
```

Anything can be represented as a *number*, i.e., data or instructions

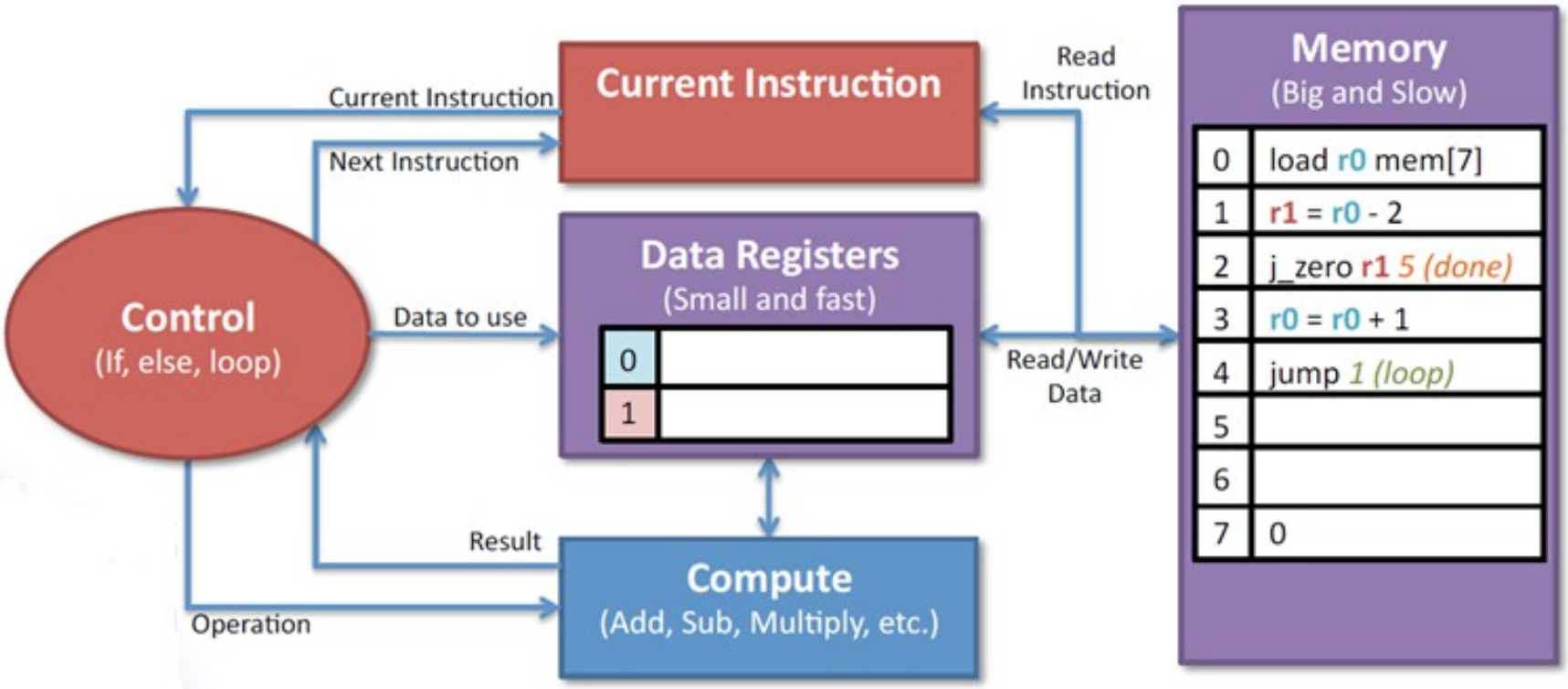
```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1                1
0101 1000 0000 1001 1100 0                1
```



Các khối xử lý cơ bản



Bộ xử lý cơ bản: Bộ nhớ, Khối điều khiển, Khối tính toán



Bộ xử lý hoạt động thế nào?

- Bộ xử lý làm gì?

– **1. Tải lệnh**

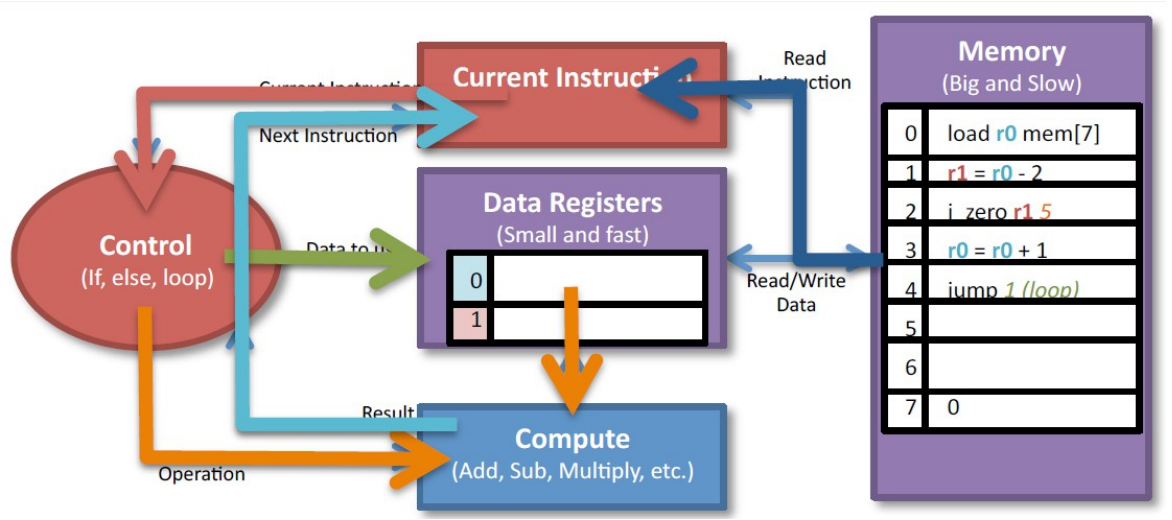
– **2. Tìm ra toán tử nào phải thực thi**

– **3. Tìm ra dữ liệu nào sử dụng**

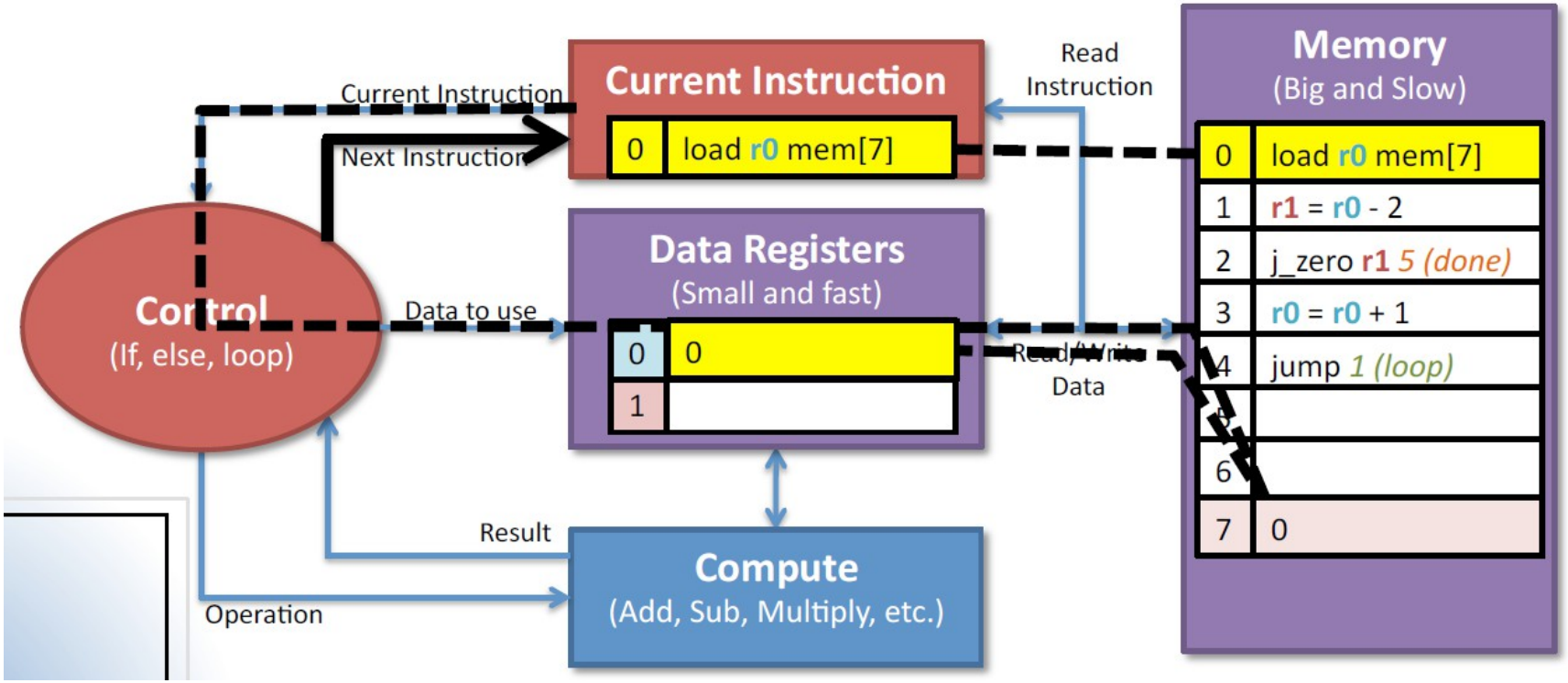
– **4. Thực hiện tính toán**

– **5. Tìm ra lệnh tiếp theo**

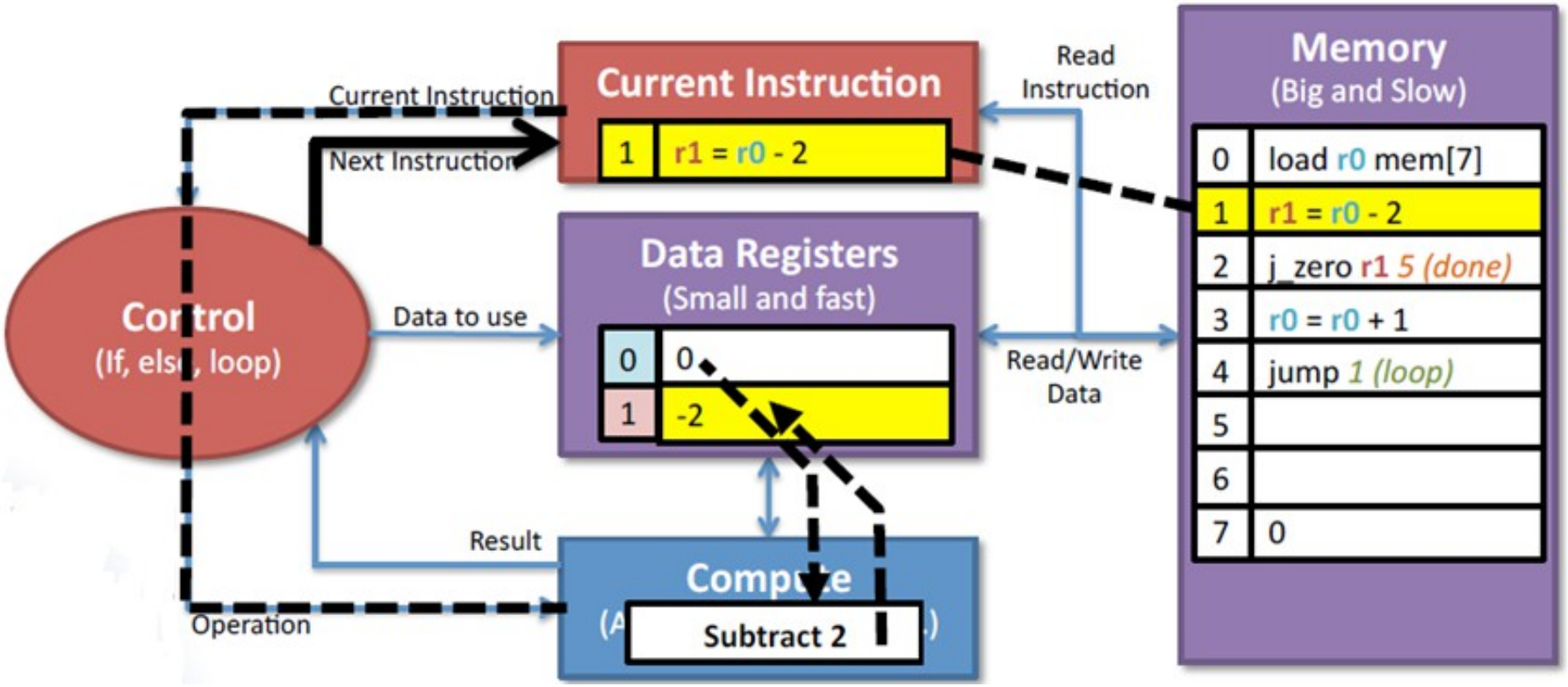
- Lặp đi lặp lại quá trình



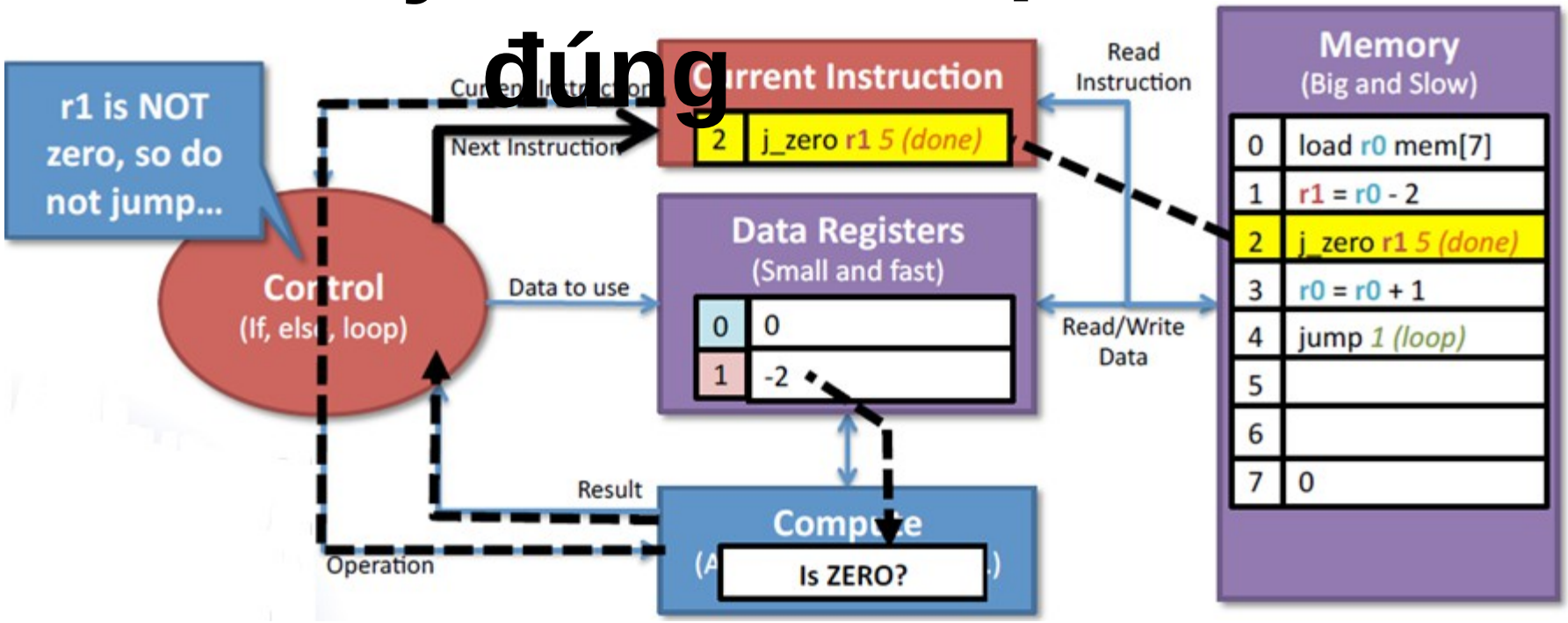
1: Tải giá trị r0 (i) từ bộ nhớ (location 7)



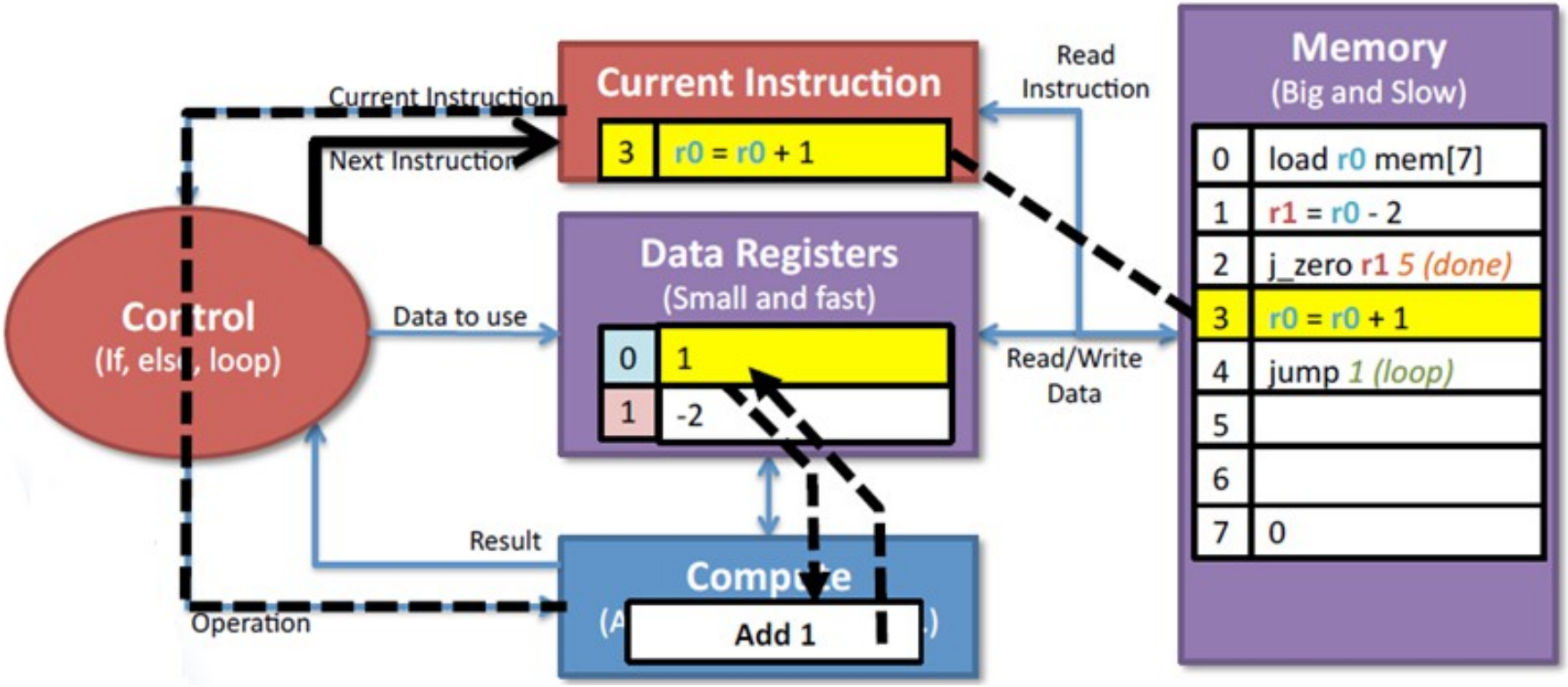
2: Trừ 2 từ r0(i)


Current Instruction
Next Instruction
Control
(If, else, loop)
Data to use
Current Instruction
1
r1 = r0 - 2
Data Registers
(Small and fast)
0
0
1
-2
Result
Compute
Subtract 2
Operation
Read
Instruction
Read/Write
Data
Memory
(Big and Slow)
0
load r0 mem[7]
1
r1 = r0 - 2
2
j_zero r1 5 (done)
3
r0 = r0 + 1
4
jump 1 (loop)
5
6
7
0

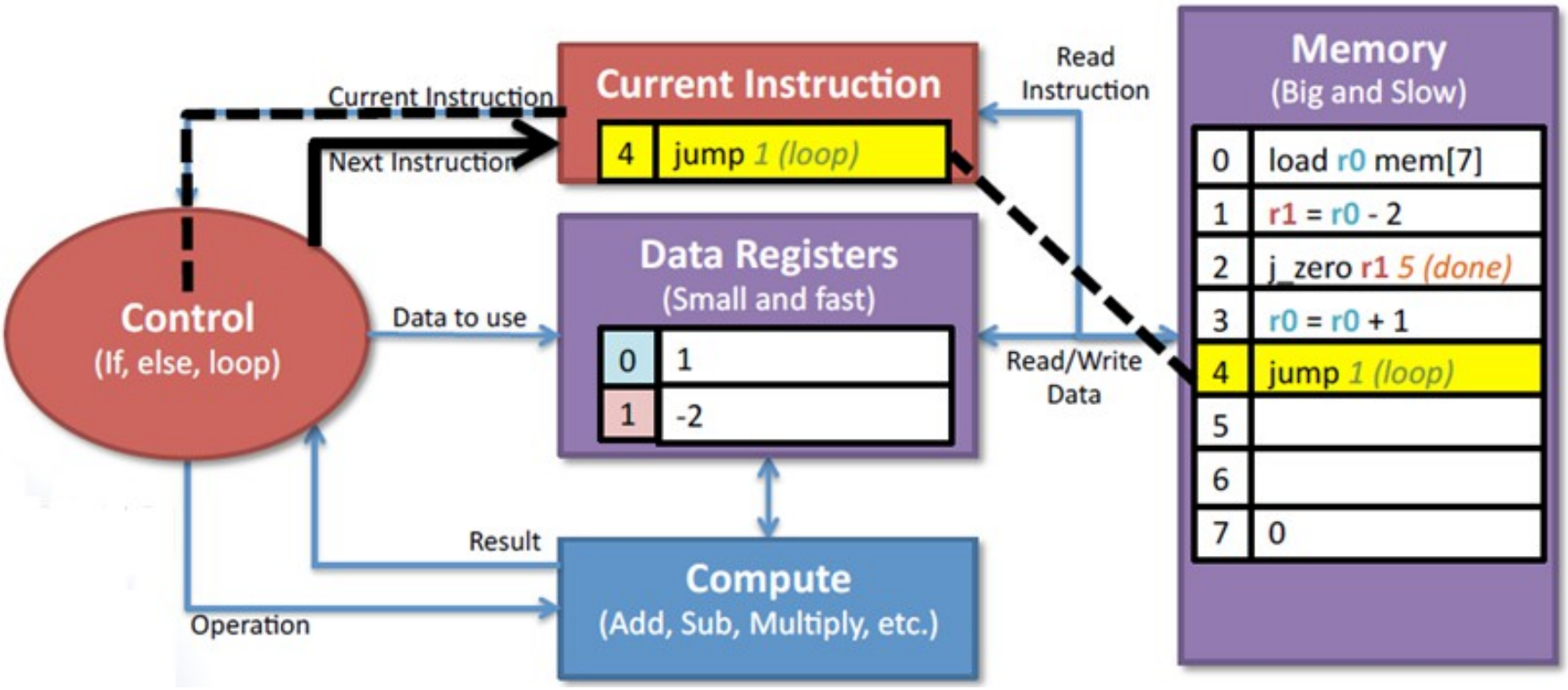
3: Kiểm tra nếu r1 bằng 0, nhảy khi điều kiện đúng



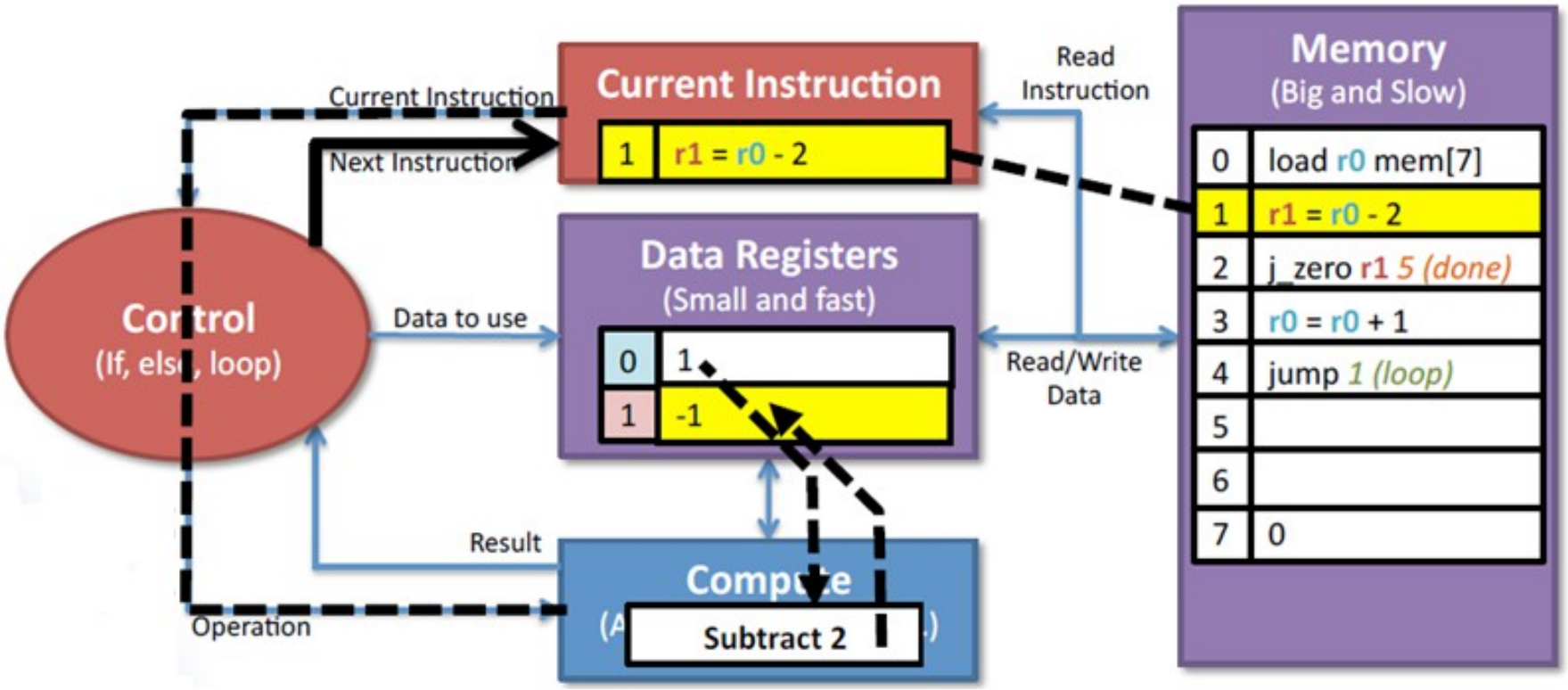
4: Tăng r0 (i)


Current Instruction
Next Instruction
Control
(If, else, loop)
Current Instruction
3
r0 = r0 + 1
Data Registers
(Small and fast)
0
1
1
-2
Data to use
Result
Operation
Compute
Add 1
Read
Instruction
Read/Write
Data
Memory
(Big and Slow)
0 load r0 mem[7]
1 r1 = r0 - 2
2 j_zero r1 5 (done)
3 r0 = r0 + 1
4 jump 1 (loop)
5
6
7 0

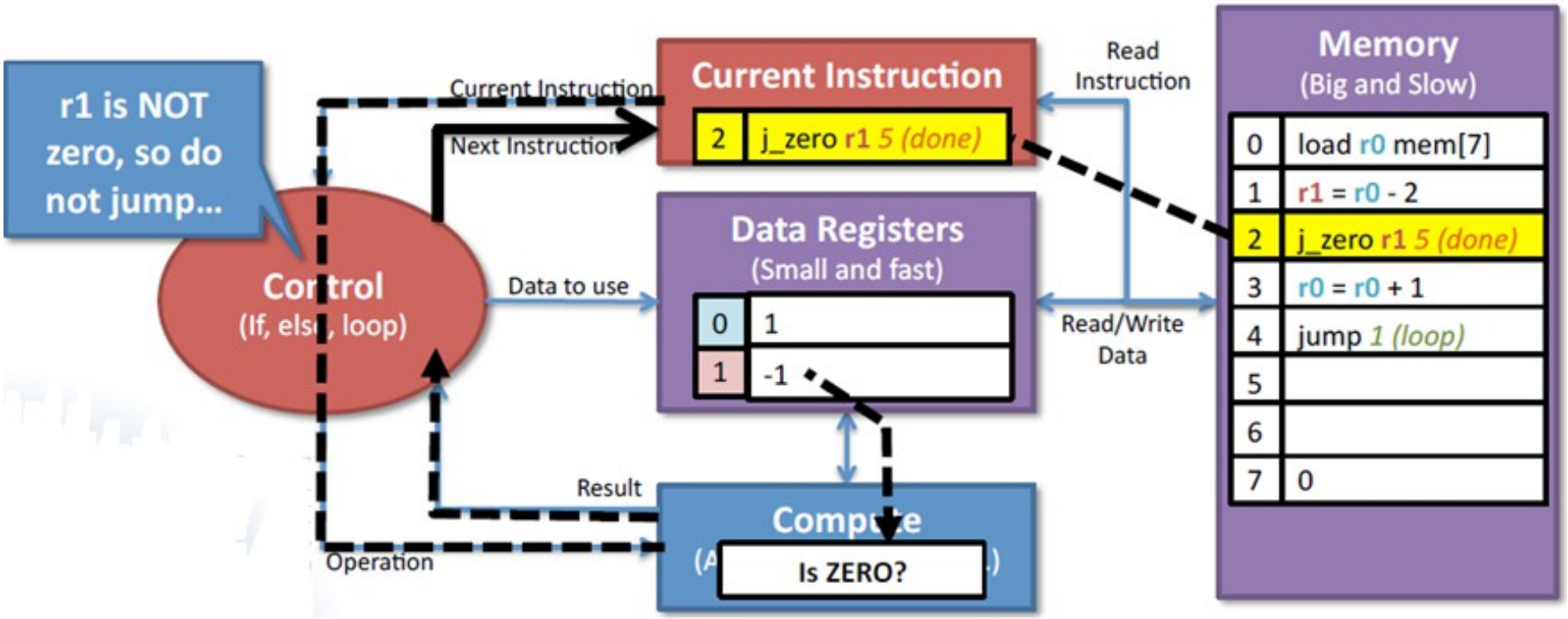
5: Tiếp tục vòng lặp



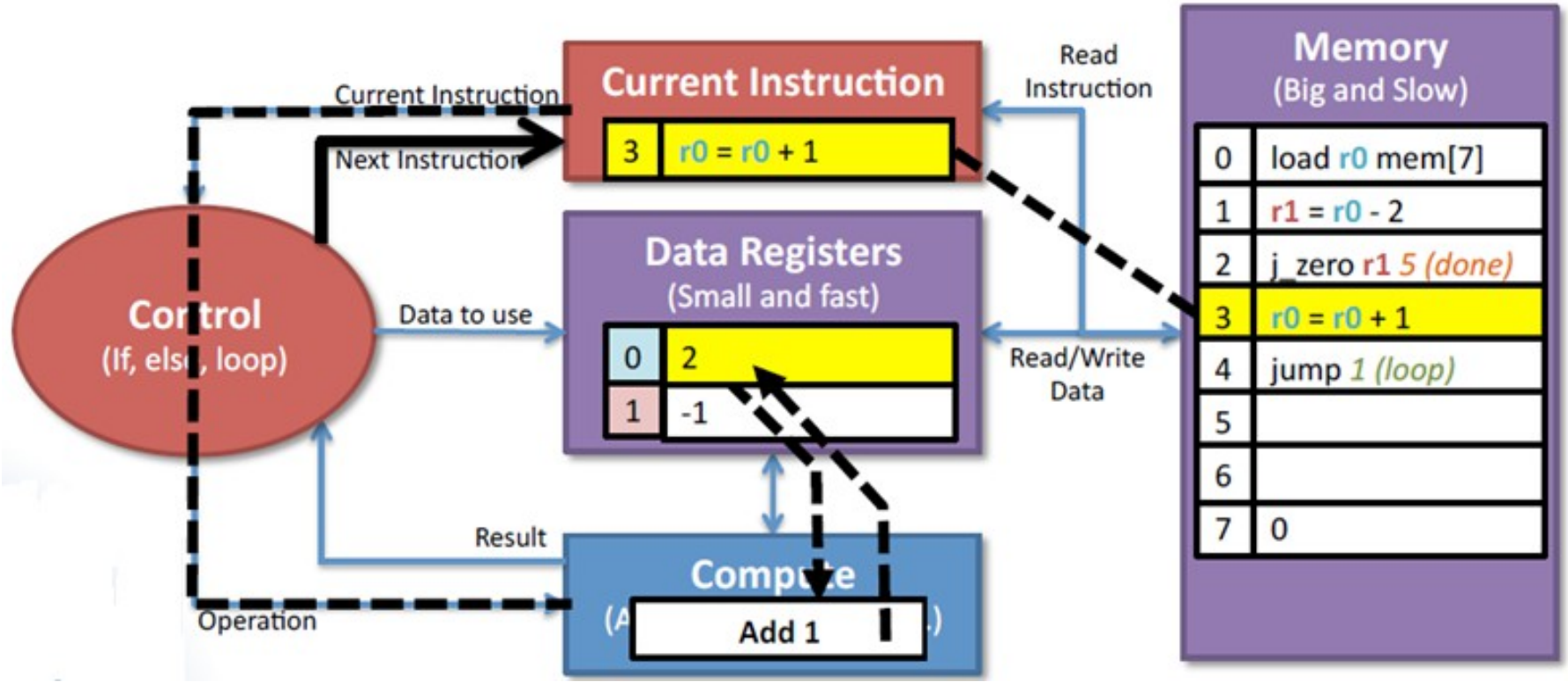
6: Trừ 2 từ r0(i)


Current Instruction
Next Instruction
Control
(If, else, loop)
Data to use
Current Instruction
1
r1 = r0 - 2
Data Registers
(Small and fast)
0
1
1
-1
Result
Operation
Compute
Subtract 2
Read
Instruction
Read/Write
Data
Memory
(Big and Slow)
0
load r0 mem[7]
1
r1 = r0 - 2
2
j_zero r1 5 (done)
3
r0 = r0 + 1
4
jump 1 (loop)
5
6
7
0

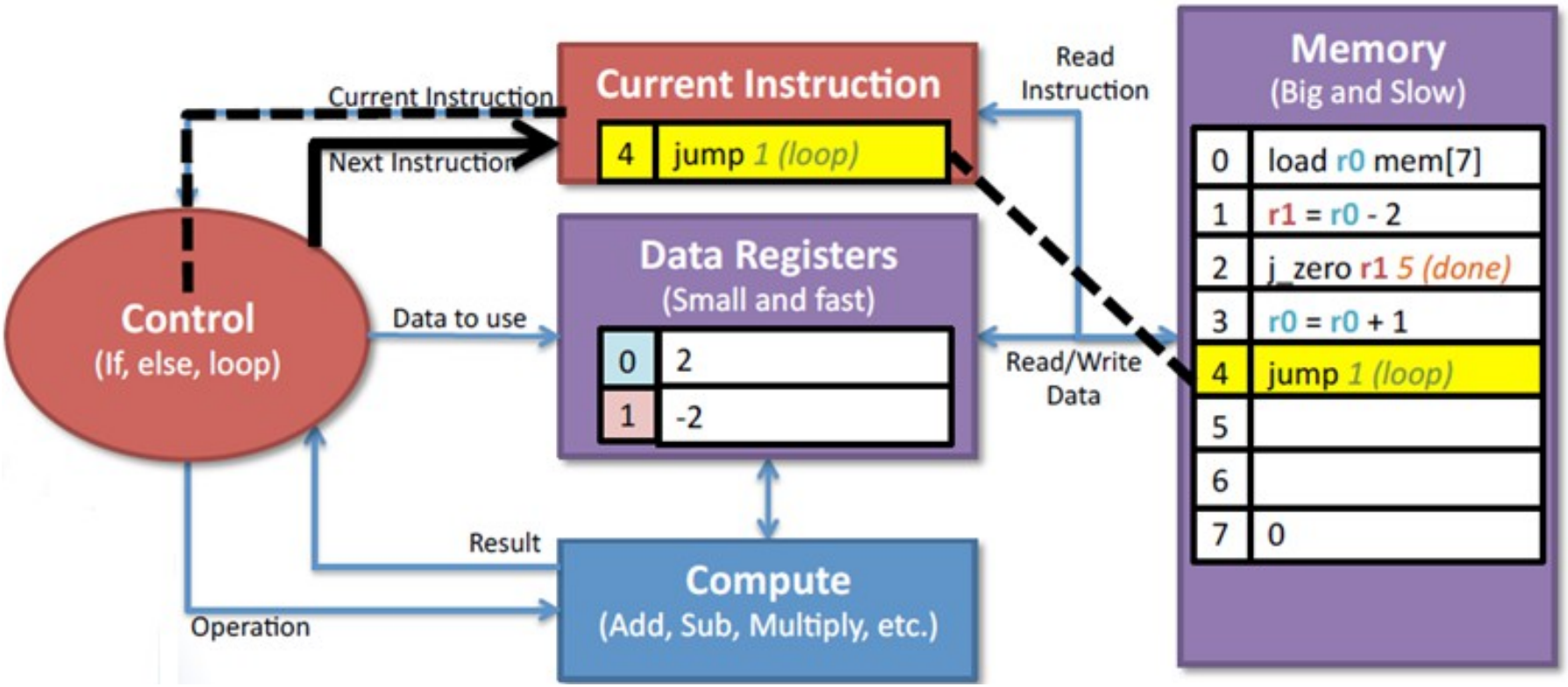
7: Kiểm tra nếu r1 bằng 0, nhảy khi điều kiện đúng



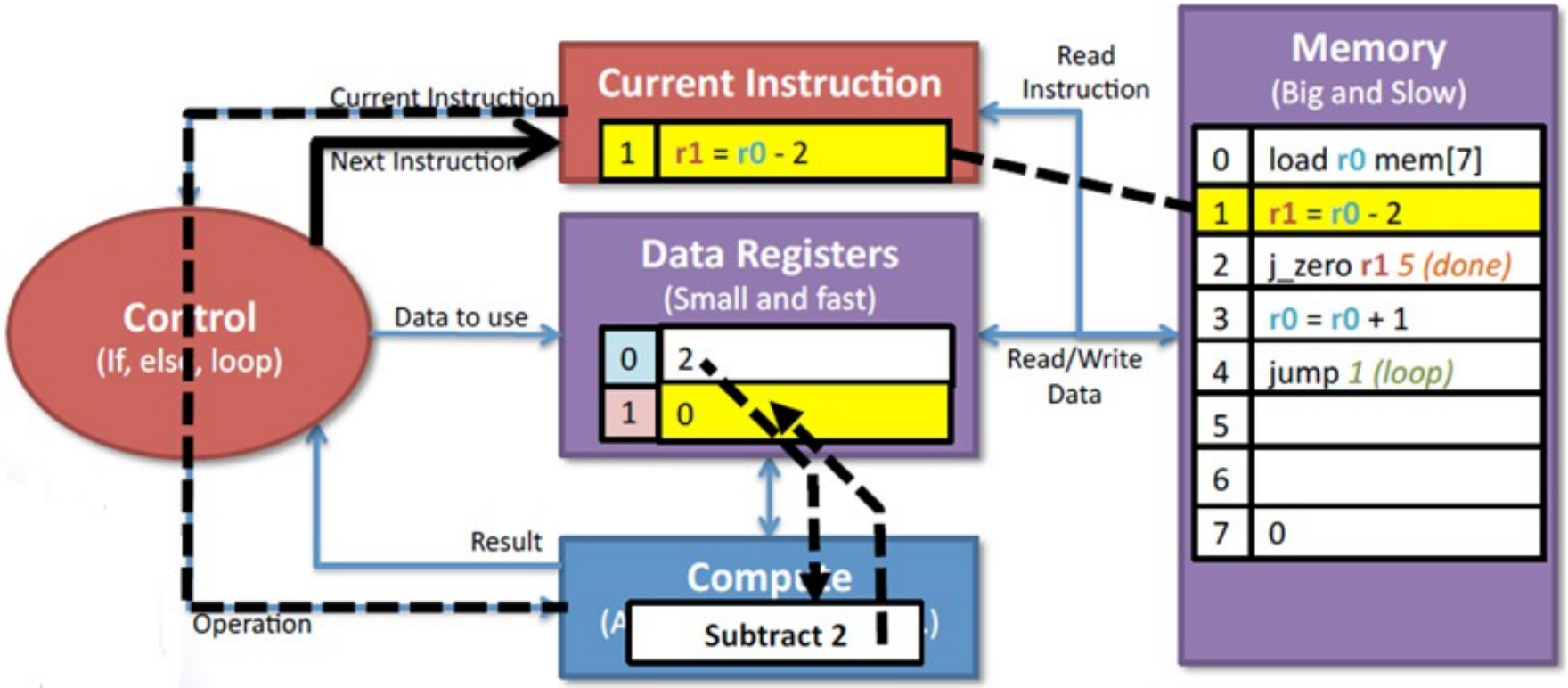
8: Tăng r0 (i)


Current Instruction
Next Instruction
Control
(If, else, loop)
Data to use
Current Instruction
3
r0 = r0 + 1
Data Registers
(Small and fast)
0
2
1
-1
Result
Operation
Add 1
Read
Instruction
Read/Write
Data
Memory
(Big and Slow)
0
load r0 mem[7]
1
r1 = r0 - 2
2
j_zero r1 5 (done)
3
r0 = r0 + 1
4
jump 1 (loop)
5
6
7
0

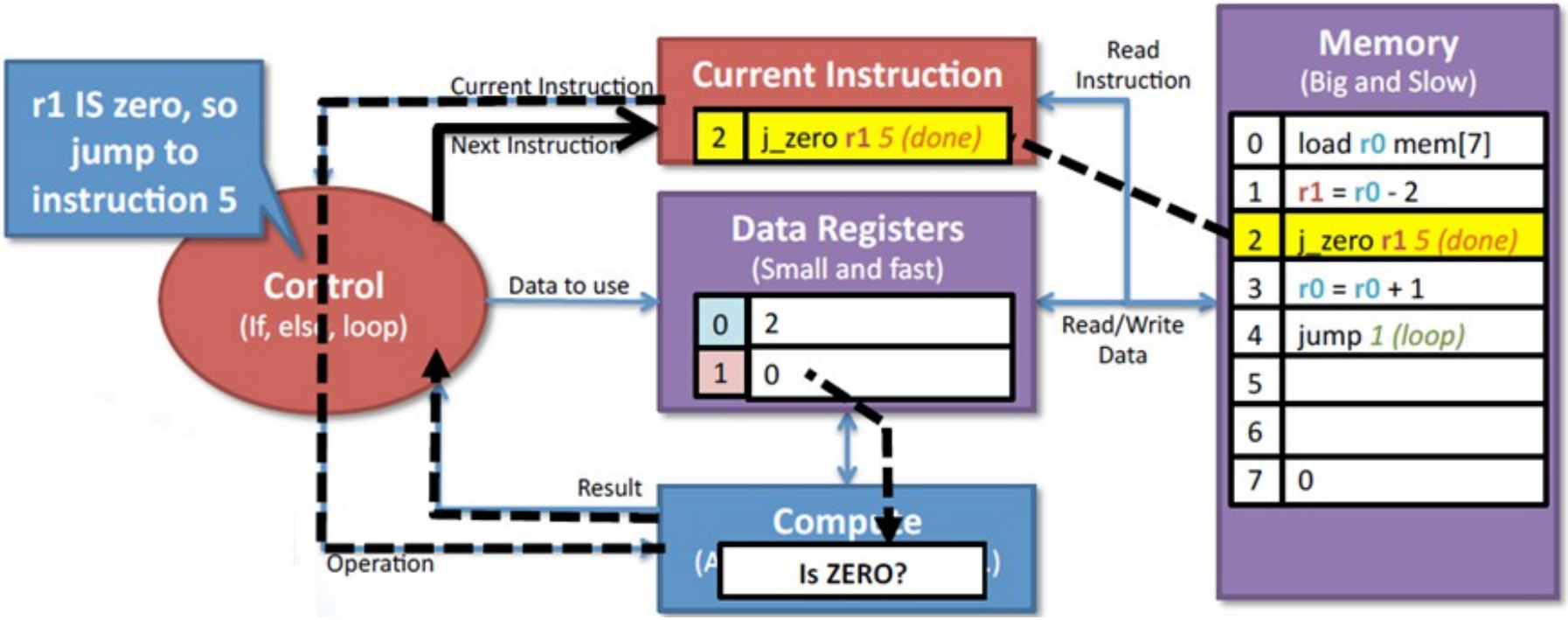
9: Tiếp tục vòng lặp



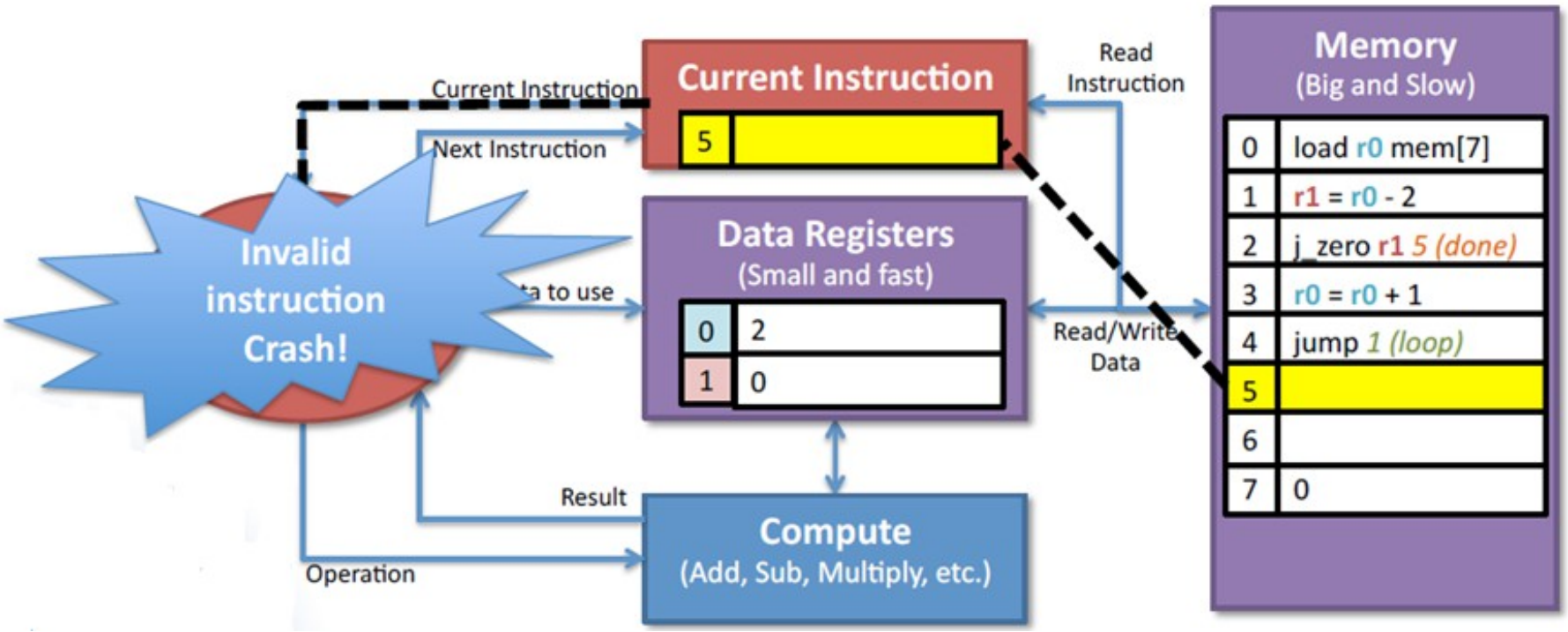
10: Trừ 2 từ r0(i)



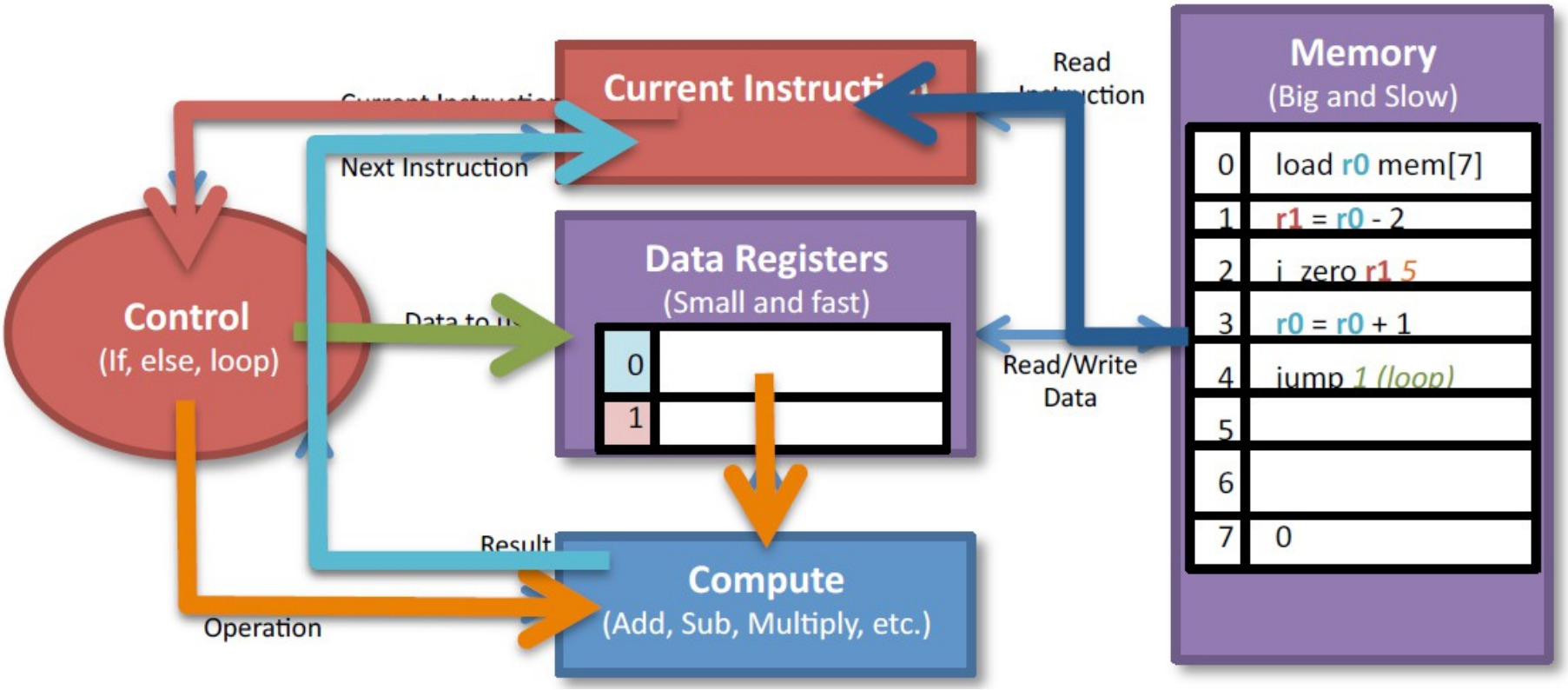
11: Kiểm tra r1 bằng 0, nhảy khi điều kiện đúng.



12: Dừng chương trình vì lệnh 5 không hợp lệ!



Hiểu chi tiết về bộ xử lý MIPS



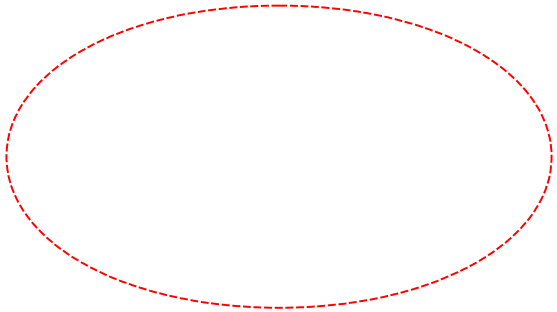
Đánh giá hiệu năng

Cách tính hiệu năng

$$\text{Performance}_x = \frac{1}{\text{Execution Time}_x}$$

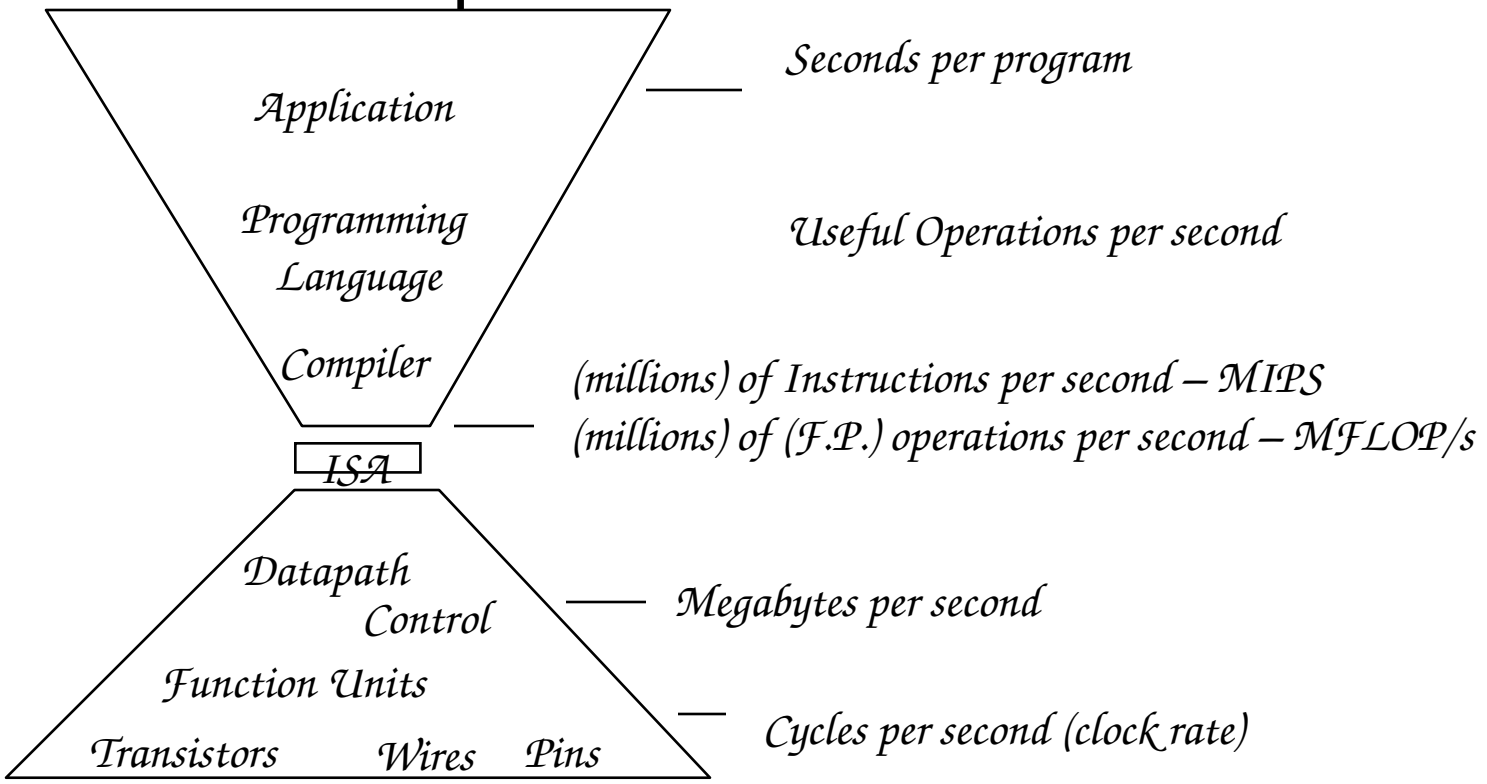
$$\frac{\text{Performance}_x}{\text{Performance}_y} \quad \frac{\text{ExecutionTime}_y}{\text{ExecutionTime}_x} \quad n$$

Cách tính hiệu năng

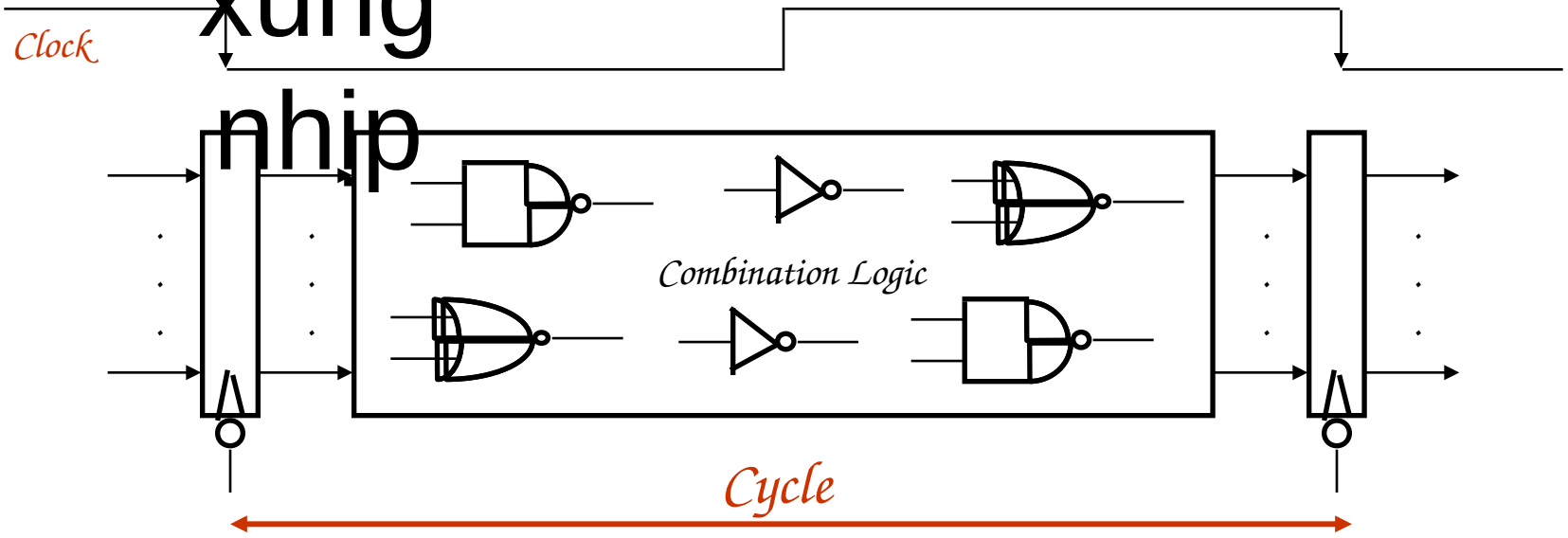


$$T_{cpu} \quad C \quad T_c \text{ or } T_{cpu} \quad C / f_c$$

Đơn vị đo hiệu năng theo các phân mức



Chu kỳ xung nhịp



Hầu hết các máy tính hiện đại không đổi và lặp đi lặp lại chu kỳ

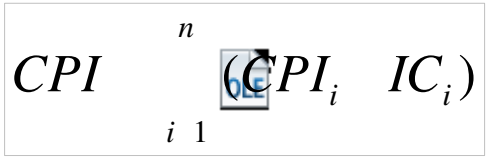
Số xung đồng hồ

- Số xung đồng hồ thực hiện 1 chương trình: $C \quad I \quad CPI$
- Trong đó:
  - *I* là số chỉ thị máy cần thực hiện trong chương trình
  - *CPI* (*eng. Clock cycles per Instruction*) là số xung đồng hồ trung bình cần để thực thi 1 chỉ thị máy,
- CPI có thể dùng để so sánh các máy tính khác nhau cùng triển khai 1 kiến trúc tập lệnh.
- Ví dụ: có 3 loại lệnh A, B, C khác nhau trong 1 kiến trúc tập lệnh. Mỗi lệnh trong từng loại có CPI tương ứng:

	CPI for this instruction class		
	A	B	C
CPI	1	2	3

CPI hiệu dụng (trung bình)

- CPI hiệu dụng được tính bằng cách xét tất cả các lớp chỉ thị có trong chương trình và lấy trung bình với trọng số là tỉ lệ xuất hiện của lớp chỉ thị trong chương trình



  - *ICi* là tỉ lệ (%) số chỉ thị thuộc loại *i* được thực thi
  - *CPIi* là số chu kỳ (trung bình) cần để thực hiện 1 chỉ thị thuộc thuộc loại *i*
  - n là số loại chỉ thị

- CPI hiệu dụng phụ thuộc vào tỉ lệ chỉ thị trong một chương trình (tần suất động của các chỉ thị trong 1 hoặc nhiều chương trình)

So sánh dựa trên CPI

- Khi 2 máy tính A, B cùng thực hiện 1 chương trình, chúng cùng thực hiện I chỉ thị.
- Do đó:

$$\begin{array}{lllllllll} T_{cpu,A} & I & CPI_A & T_{c,A} & I & 2{,}0 & 250\,ps & 500 & I \\ T_{cpu,B} & I & CPI_B & T_{c,B} & I & 1{,}2 & 500\,ps & 600 & I \end{array}$$

- Máy A nhanh hơn máy B:

$$\frac{\text{Performance}_A}{\text{Performance}_B} = \frac{T_{cpu,B}}{T_{cpu,A}} = \frac{600 \times I}{500 \times I} = 1{,}2$$

Máy tính A và B cùng triển khai 1 kiến trúc tập lệnh. Máy A có chu kỳ đồng hồ là 250ps, và CPI hiệu dụng cho 1 chương trình P là 2,0. Máy B có chu kỳ đồng hồ là 500ps, và CPI hiệu dụng cho cùng 1 chương trình P là 1,2. Máy tính nào nhanh hơn và nhanh hơn bao nhiêu?

Máy tính A với xung đồng hồ 2GHz thực hiện 1 chương trình hết 10 giây. Để thực hiện chương trình đó trong 6 giây bằng máy tính B, ta cần tăng tốc độ xung đồng hồ của máy B. Tuy nhiên, tăng tốc độ xung đồng hồ cũng làm tăng số chu kỳ cần thiết lên 1,2 lần. Xác định tốc độ xung đồng hồ máy tính B

- Công thức để tính thời gian CPU, khi máy tính A thực hiện chương trình:

$$T_{cpu,A} = \frac{C_A}{f_{c,A}}$$

- Số chu kỳ máy tính A dùng để thực hiện chương trình:

$$C_A = T_{cpu,A} \times f_{c,A} = 10 \times 2 \times 10^9 = 20 \times 10^9 \text{ cycles}$$

$$C_B = 1{,}2 \times C_A$$

- Số chu kỳ máy tính B dùng để thực hiện chương trình:

$$f_{c,B} = \frac{C_B}{T_{cpu,B}} = \frac{1{,}2 \times 20 \times 10^9}{6} = 4GHz$$

- Tốc độ đồng hồ của máy tính B:

So sánh đoạn mã chương trình

- Người thiết kế một máy tính triển khai kiến trúc tập lệnh gồm 3 loại chỉ thị A, B, C được CPI như sau:

	A	B	C
CPI	1	2	3

- Với [illegible] chọn

Đoạn mã	A	B	C
1	2	1	2
2	4	1	1

So sánh đoạn mã chương trình

Đoạn mã	A	B	C
1	2	1	2
2	4	1	1

	A	B	C
CPI	1	2	3

- Đoạn mã 1 dùng 5 chỉ thị, đoạn mã 2 dùng 6 chỉ thị
- Số xung đồng hồ để thực hiện mỗi đoạn mã được tính như sau:

$$C_1 = \sum_{i=1}^{3} (CPI_i \times I_{1,i}) = (1 \times 2 + 2 \times 1 + 3 \times 2) = 10$$

$$C_2 = \sum_{i=1}^{3} (CPI_i \times I_{2,i}) = (1 \times 4 + 2 \times 1 + 3 \times 1) = 9$$

- Trong đó *I1,i, I2,i* là số lượng chỉ thị loại *i* trong đoạn mã 1 và 2 tương ứng

Các yếu tố ảnh hưởng đến hiệu năng

$$CPUtime = \frac{Seconds}{Program} = \frac{Cycles}{Program} \cdot \frac{Seconds}{Cycle}$$

$$= \frac{Instructions}{Program} \cdot \frac{Cycles}{Instruction} \cdot \frac{Seconds}{Cycle}$$

	IC	CPI	Clock Cycle
Program	√		
Compiler	√	(√)	
Instruction Set	√	√	
Organization		√	√
Technology			√