CHƯƠNG 1: CÁC VẤN ĐỀ CƠ BẢN TRONG THIẾT KẾ MÁY TÍNH

➢ Công thức tính hiệu suất:

$Performance = \frac{1}{Excution\ time}$ (***Performance***: *Hiệu suất*, ***Excution time***: *Thời gian thực thi*)

Nếu máy tính A nhanh hơn máy tính B n lần thì ta có mối quan hệ giữa hai hiệu suất máy tính A và B là:

$\frac{Performance_A}{Performance_B} = \frac{Excution\ time_B}{Excution\ time_A} = n$

➢ Công thức tính thời gian xử lý CPU:

$$Clock\ cycle\ time = \frac{1}{Clock\ rate}$$

$$CPU\ time = CPU\ clock\ cycles \times Clock\ cycle\ time = IC \times CPI \times Clock\ cycle\ time = \frac{IC \times CPI}{Clock\ rate}$$

CPU time: Thời gian CPU xử lý một chương trình

CPU clock cycles: Tổng số chu kỳ xung nhịp cần thiết để thực thi chương trình

Clock cycle: Thời gian của 1 chu kỳ xung nhịp

Clock rate (Clock frequency): Tần số xung nhịp

IC (Instruction count): Số lệnh của chương trình

CPI (Clock cycles per instruction): Số chu kỳ cần thiết để hoàn thành lệnh

Các loại lệnh khác nhau thực hiện với số chu kỳ khác nhau trên mỗi lệnh: $CPU\ Clock\ cycles = \sum_{i=1}^{n}(CPI_i \times IC_i)$

CPI trung bình: $CPI = \frac{CPU\ clock\ cycles}{IC} = \frac{\sum_{i=1}^{n}(CPI_i \times IC_i)}{IC}$

CHƯƠNG 2: KIẾN TRÚC TẬP LỆNH

➢ Biểu diễn số nhị phân

Cho 1 số n bit, có dạng: $x = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_1 2^1 + x_0 2^0$

- Số nguyên nhị phân không dấu: tầm vực giá trị sẽ là 0 đến $+2^n - 1$. Ví dụ giá trị 1 số nhị phân không dấu 32 bit sẽ nằm trong phạm vi từ 0 đến +4.294.967.295
- Số nguyên nhị phân có dấu dạng bù 2: tầm vực giá trị sẽ là -2^{n-1} đến $+2^{n-1} - 1$. Ví dụ giá trị 1 số nhị phân có dấu 32 bit sẽ nằm trong phạm vi từ -2.147.483.648 đến +2.147.483.647

➢ Cách xác định 1 từ nhớ (word)

- Địa chỉ 1 từ nhớ (word) sẽ là địa chỉ của ô nhớ thấp nhất trong từ nhớ đó và địa chỉ này phải chia hết cho 4
- Ô nhớ được truy xuất sẽ có địa chỉ tuyệt đối được tính bằng công thức: **Memory location = <offset> + <base register>**

➢ Biểu diễn lệnh

Thanh ghi	Chỉ số	Mục đích sử dụng
$zero	0	Hằng số 0 (không thể thay đổi), và là thanh ghi chỉ đọc
$at	1	Dành riêng cho trình hợp dịch (Assembler)
$v0 - $v1	2 - 3	Dùng chứa dữ liệu trả về (returned values) của hàm
$a0 - $a3	4 - 7	Dùng truyền tham số (parameters) cho các hàm
$t0 - $t7	8 - 15	Dùng lưu trữ các dữ liệu tạm thời
$s0 - $s7	16 - 23	Dùng lưu trữ các biến dữ liệu
$t8 - $t9	24 - 25	Dùng lưu trữ các dữ liệu tạm thời
$k0 - $k1	26 - 27	Dành riêng cho hệ điều hành
$gp	28	Con trỏ toàn cục (global pointer)
$sp	29	Con trỏ chồng dữ liệu (stack pointer)
$fp	30	Con trỏ khung dữ liệu (frame pointer)
$ra	31	Địa chỉ trả về (returned address) và chỉ được cập nhật bởi phần cứng

Lệnh dạng R:

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

Cấu trúc thành phần của lệnh dạng **R**:

- op: Mã lệnh (opcode) giá trị luôn là 0
- rs: Chỉ số thanh ghi nguồn thứ nhất
- rt: Chỉ số thanh ghi nguồn thứ nhì
- rd: Chỉ số thanh ghi đích
- shamt: Số bit dịch chuyển
- funct: mã chức năng (xem bảng các giá trị trường funct của lệnh dạng **R**)

```
add $t0, $s1, $s2
```

special	$s1	$s2	$t0	0	add
0	17	18	8	0	32
000000	10001	10010	01000	00000	100000

$00000010001100100100000000100000_2 = 02324020_{16}$

Lệnh dạng I:

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

Cấu trúc thành phần của lệnh dạng **I**:

- op: Mã lệnh (xem bảng các giá trị trường opcode của lệnh dạng **I**)
- rs: Chỉ số thanh ghi nguồn thứ nhất hoặc chỉ số thanh ghi địa chỉ
- rt: Chỉ số thanh ghi nguồn thứ nhì hoặc thanh ghi dữ liệu
- constant/address: biểu diễn giá trị số nguyên trong các lệnh có sự tham gia của số nguyên

lw $t0, 32($s3)

lw	$s3	$t0	32
35	19	8	32
100011	10011	01000	0000000000100000

Lệnh dạng J:

Opcode	address
6 bit	26 bit

Cấu trúc thành phần của lệnh dạng **J**:

- op: Mã lệnh (xem bảng các giá trị trường opcode của lệnh dạng **J**) , chỉ có 2 lệnh dạng J là **j** và **jal**
- address: đĩa chỉ của nhãn trong lệnh rẻ nhánh

j label (nhãn label có địa chỉ là 20000)

j	label
2	20000
000010	00000000000100111000100000

Giá trị trường ***funct*** của các lệnh có định dạng **R**:

Lệnh	funct	Lệnh	funct
add	32	srl	2
and	36	sltu	43
nor	39		
sll	0		
slt	42		
sub	34		
or	37		
jr	8		

Giá trị trường ***opcode*** của các lệnh có định dạng **I**:

Lệnh	opcode	Lệnh	opcode
addi	8	addiu	9
lbu	36	lhu	37
lb	32	lh	32
lw	35	sw	43
sb	40	sh	41
slti	10	sltiu	11
andi	12	ori	13
beq	4	bne	5

Giá trị trường ***opcode*** của các lệnh có định dạng **J**:

Lệnh	opcode
j	2
jal	3

➢ Đổi hệ cơ số

1. Chuyển đổi từ hệ cơ số thập phân sang nhị phân và ngược lại (DEC <->BIN).

➢ Từ thập phân sang nhị phân.

Đầu tiên chúng ta chia số cần chuyển cho 2 và lấy phần dư, rồi tiếp tục chia phần nguyên lấy phần dư, sau đó sắp xếp thứ tự phần dư theo thứ tự ngược từ dưới lên.

VD: Chuyển 2371 (hệ thập phân) sang hệ nhị phân?

2371 chia 2 = 1185.5 (1185 -> dư 1)
1185 chia 2 = 592 -> dư 1
(phần nguyên)
592 chia 2 = 296 -> dư 0
296 chia 2 = 148 -> dư 0
148 chia 2 = 74 -> dư 0
74 chia 2 = 37 -> dư 0
37 chia 2 = 18 -> dư 1
18 chia 2 = 9 -> dư 0
9 chia 2 = 4 -> dư 1
4 chia 2 = 2 -> dư 0
2 chia 2 = 1 -> dư 0
1 chia 2 = 0 -> dư 1

Sắp xếp thứ tự số dư từ dưới lên trên: $2371_{DEC} = 100101000011_{BIN}$

➢ Từ nhị phân sang thập phân.

Muốn chuyển đổi cơ số từ hệ nhị phân sang thập phân, ta lấy các chữ số trong phần nguyên của số cần chuyển nhân lần lượt với 2 mũ 0,1,2,3,…tăng dần từ phải qua trái. Còn phần nguyên của số cần chuyển ta sẽ nhân lần lượt với 2 mũ -1, -2, -3, … giảm dần từ phải qua trái. Phần nguyên và phần thập phân được ngăn cách nhau bằng dấu chấm “.”

2. Chuyển đổi từ hệ cơ số thập phân sang bác phân và ngược lại (DEC <->OCT).

➢ Từ thập phân sang bác phân.

Cũng giống như cách chuyển đổi cơ số từ thập phân sang nhị phân, để chuyển từ thập phân sang bác phân ta cũng chia số cần chuyển cho 8 được phần dư (giá trị dư từ 1->7), sau đó cũng lấy phần nguyên chia tiếp và lấp phần dư, kết quả là phần dư được sắp xếp theo thứ tự từ dưới lên trên.

VD: Chuyển số 2764 (hệ thập phân) sang hệ bác phân?

2764 chia 8 = 345.5 (345 -> dư 4(lấy phần lẻ nhân với 8))
345 chia 8 = 43.125 (43 -> dư 1)
43 chia 8 = 5.375 (5 -> dư 3)
5 chia 8 = 0 -> dư 5

Sắp xếp thứ tự từ dưới lên trên: $2764_{DEC} = 5314_{OCT}$

➢ Từ bác phân sang thập phân.

Tương tự hệ nhị phân, để chuyển đổi cơ số từ hệ bác phân sang thập phân, ta lấy các chữ số trong phần nguyên của số cần chuyển nhân lần lượt với 8 mũ 0,1,2,3,…tăng dần từ phải qua trái. Còn phần nguyên của số cần chuyển ta sẽ nhân lần lượt với 8 mũ -1, -2, -3, … giảm dần

3. Chuyển đổi từ hệ cơ số thập phân sang thập lục phân và ngược lại (DEC <->HEX).

➢ Từ thập phân sang thập lục phân phân.

Việc chuyển đổi này cũng tương tự như nhị phân và bác phân. Cụ thể ta xét ví dụ sau đây: (sử dụng bảng 1)

3295 chia 16 = 205.9375 (205 -> dư 15) tức là chữ F
205 chia 16 = 12.8125 (12 -> dư 13) tức là D
12 chia 16 = 0 (dư 12) tức là C

Vậy $3295_{DEC} = CDF_{HEX}$

➢ Từ thập lục phân sang thập phân.

Tương tự ta nhân từng số với 16 mũ ……

VD: $CDF.91_{HEX} = C.16^2 + D.16^1 + F.16^0 + 9.16^{-1} + 1.16^{-2}$
$= 12.16^2 + 13.16^1 + 15.16^0 + 9.16^{-1} + 1.16^{-2}$
$= 3072 + 208 + 15 + 0.5625 + 0.00390625$
$= 3295.56640625$

Vậy $CDF.91_{HEX} = 3295.56640625_{DEC}$

4. Các chuyển đổi khác

➢ Từ nhị phân sang bác phân

Để chuyển đổi cơ số từ hệ nhị phân sang bác phân ta gom 3 chữ số của số cần chuyển theo thứ tự lần lượt từ phải sang trái, sau đó sử dụng bảng 1 để chuyển đổi thành kết quả mong muốn.

VD:

100110001011010_{BIN} = 100 110 001 011 010
= 4 6 1 3 2

Vậy $100110001011010_{BIN} = 46132_{OCT}$

➢ Từ nhị phân sang thập lục phân

Tương tự như trên,muốn chuyển đổi từ hệ nhị phân sang thập lục phân, ta gom 4 chữ số của số cần chuyển theo thứ tự lần lượt từ phải sang trái, sau đó sử dụng bảng 1.

VD:

100110001011010_{BIN} = 0100 1100 0101 1010 (nếu các số cuối cùng bên trái không đủ 4 chữ số thì mặc định ta thêm vào trước đó các chữ số 0)
= 4 C 5 A

Vậy $100110001011010_{BIN} = 4C5A_{HEX}$

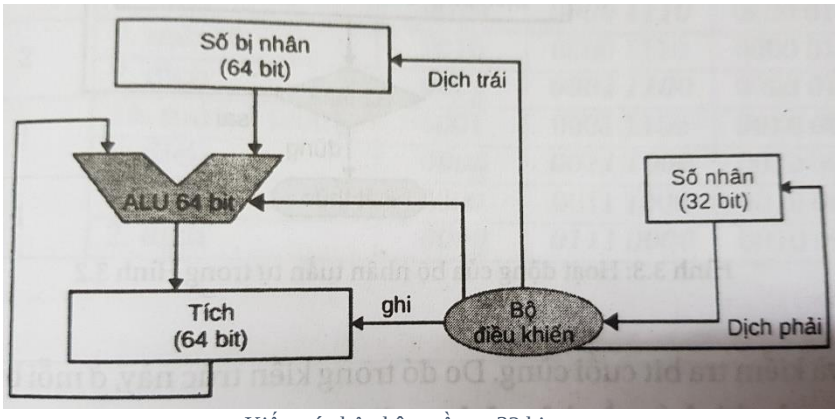
➢ Từ bác phân sang thập lục phân và ngược lại

Muốn chuyển từ hệ bác phân sang hệ thập lục phân hoặc từ thập lục phân sang bác phân, trước tiên ta phải chuyển số cần chuyển sang hệ cơ số 2 (hệ nhị phân), sau đó mới chuyển sang hệ thập lục phân hay bác phân theo các bước phía trên.

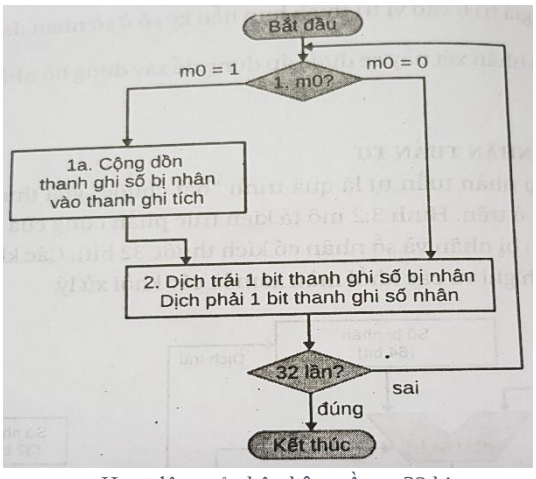
VD:

46132_{OCT} = 100 110 001 011 010_{BIN} = 0100 1100 0101 1010_{BIN}
$= 4C5A_{HEX}$

➢ Bộ nhân tuần tự



Kiến trúc bộ nhân tuần tự 32 bit

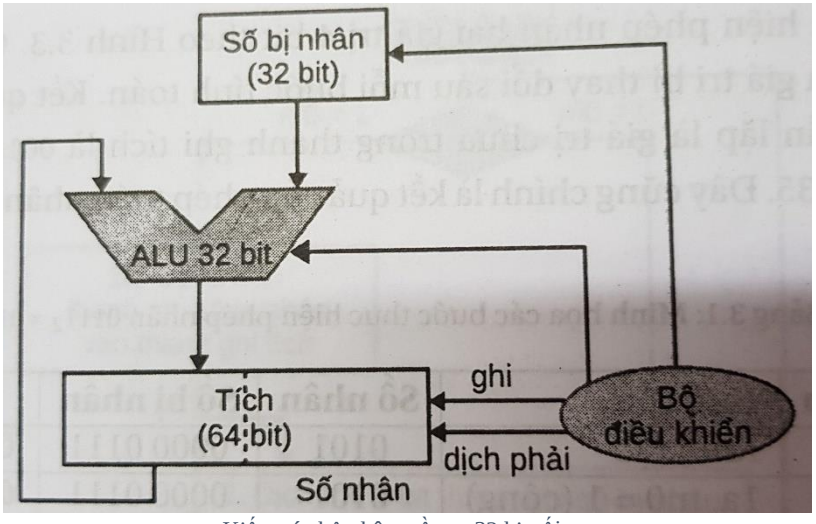


Hoạt động của bộ nhân tuần tự 32 bit

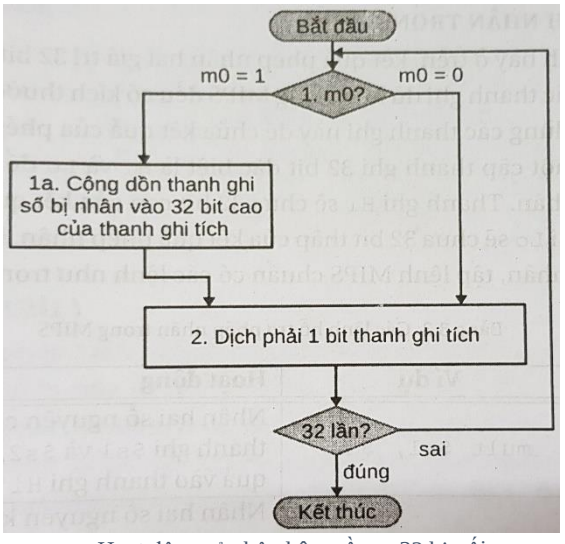
Lần lặp	Bước	Số nhân	Số bị nhân	Tích
0	Khởi tạo	0101	0000 0111	0000 0000
1	1a. m0 = 1 (cộng)	0101	0000 0111	***0000 0111***
	2. dịch	***0010***	***0000 1110***	0000 0111
2	1. m0 = 0	0010	0000 1110	0000 0111
	2. dịch	***0001***	***0001 1100***	0000 0111
3	1a. m0 = 1 (cộng)	0001	0001 1100	***0010 0011***
	2. dịch	***0000***	***0011 1000***	0010 0011
4	1. m0 = 0	0000	0011 1000	0010 0011
	2. dịch	***0000***	***0111 0000***	0010 0011

Minh họa các bước thực hiện phép nhân $\mathbf{0111_2 \times 0101_2}$ sử dụng bộ nhân tuần tự

➢ Bộ nhân tuần tự tối ưu



Kiến trúc bộ nhân tuần tự 32 bit tối ưu

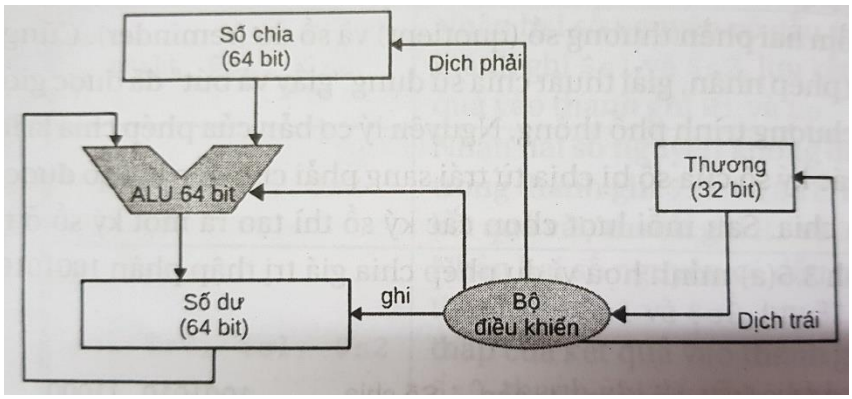


Hoạt động của bộ nhân tuần tự 32 bit tối ưu

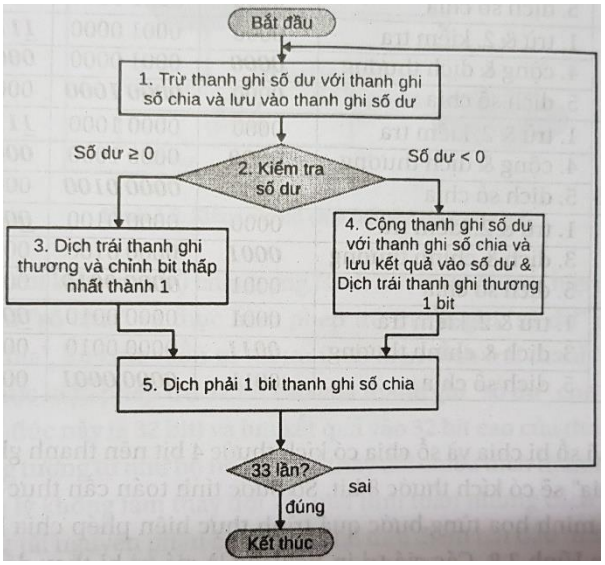
Lần lặp	Bước	Số bị nhân	Tích
0	Khởi tạo	0111	0000 0101
1	1a. m0 = 1 (cộng)	0111	***0111 0101***
	2. dịch	0111	**0011 1010**
2	1. m0 = 0	0111	0011 1010
	2. dịch	0111	**0001 1101**
3	1a. m0 = 1 (cộng)	0111	***1000 1101***
	2. dịch	0111	**0100 0110**
4	1. m0 = 0	0111	0100 0110
	2. dịch	0111	**0010 0011**

Minh họa các bước thực hiện phép nhân $\mathbf{0111_2 \times 0101_2}$ sử dụng bộ nhân tuần tự tối

➢ Bộ chia tuần tự



Kiến trúc bộ chia tuần tự 32 bit

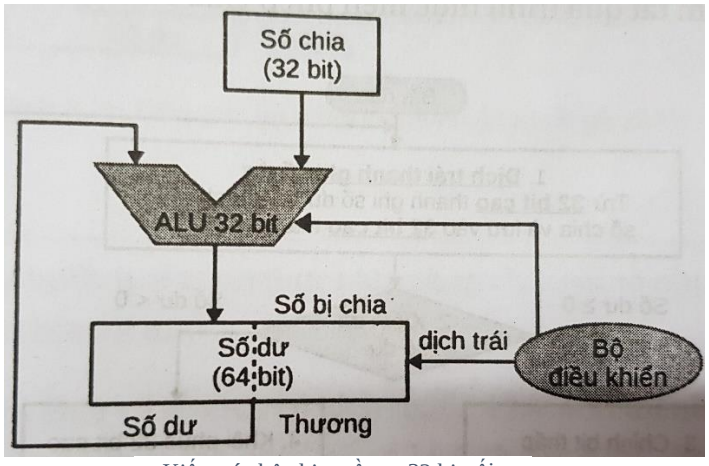


Hoạt động của bộ chia tuần tự 32 bit

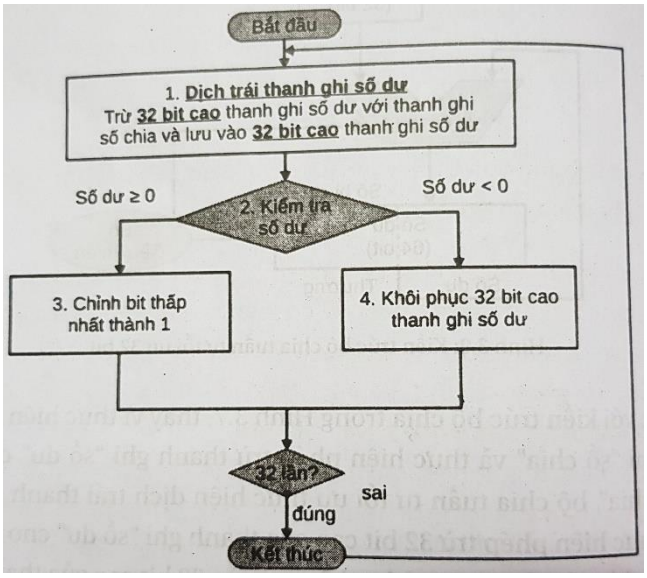
Lần lặp	Bước	Thương	Số chia	Số dư
0	Khởi tạo	0000	0010 0000	0000 0111
1	1. trừ & 2. kiểm tra	0000	0010 0000	***1110 0111***
	4. cộng & dịch thương	***0000***	0010 0000	***0000 0111***
	5. dịch số chia	0000	***0001 0000***	0000 0111
2	1. trừ & 2. kiểm tra	0000	0001 0000	***1111 0111***
	4. cộng & dịch thương	***0000***	0001 0000	***0000 0111***
	5. dịch số chia	0000	***0000 1000***	0000 0111
3	1. trừ & 2. kiểm tra	0000	0000 1000	***1111 1111***
	4. cộng & dịch thương	***0000***	0000 1000	***0000 0111***
	5. dịch số chia	0000	***0000 0100***	0000 0111
4	1. trừ & 2. kiểm tra	0000	0000 0100	***0000 0011***
	3. dịch & chỉnh thương	***0001***	0000 0100	0000 0011
	5. dịch số chia	0001	***0000 0010***	0000 0111
5	1. trừ & 2. kiểm tra	0001	0000 0010	***0000 0001***
	3. dịch & chỉnh thương	***0011***	0000 0010	0000 0001
	5. dịch số chia	0011	***0000 0001***	0000 0001

Minh họa các bước thực hiện phép chia **0111_2 ÷ 0010_2** sử dụng bộ chia tuần tự

➢ Bộ chia tuần tự tối ưu



Kiến trúc bộ chia tuần tự 32 bit tối ưu

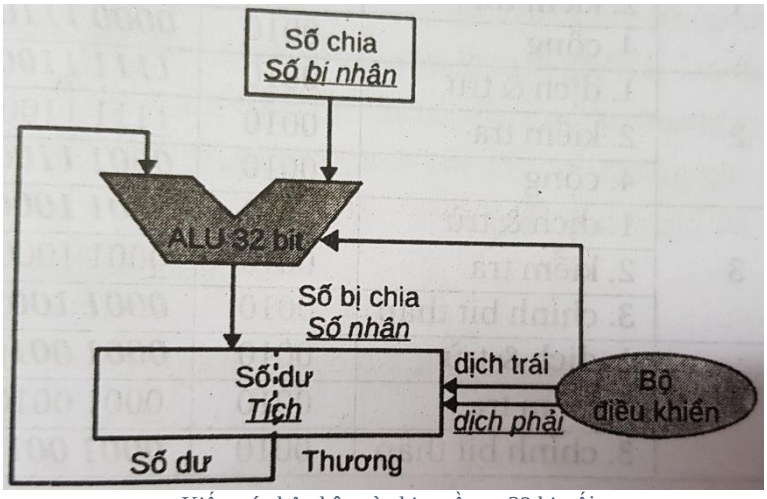


Hoạt động của bộ chia tuần tự 32 bit tối ưu

Lần lặp	Bước	Số chia	Số dư
0	Khởi tạo	0010	0000 0111
1	1. dịch & trừ	0010	***1110 1110***
	2. kiểm tra	0010	1110 1110
	4. cộng	0010	***0000 1110***
2	1. dịch & trừ	0010	***1111 1100***
	2. kiểm tra	0010	1111 1100
	4. cộng	0010	***0001 1100***
3	1. dịch & trừ	0010	***0001 1000***
	2. kiểm tra	0010	0001 1000
	3. chỉnh bit thấp	0010	***0001 1001***
4	1. dịch & trừ	0010	***0001 0010***
	2. kiểm tra	0010	0001 0010
	3. chỉnh bit thấp	0010	***0001 0011***

Minh họa các bước thực hiện phép chia **0111_2 ÷ 0010_2** sử dụng bộ chia tuần tự tối ưu

➢ Kiến trúc phần cứng dùng chung cho cả 2 phép toán nhân và chia



Kiến trúc bộ nhân và chia tuần tự 32 bit tối ưu

➢ Biểu diễn số thực dấu chấm động dạng chính xác đơn 32 bit theo chuẩn IEEE 754

1 bit	8 bits	23 bits
S	Exponent	Fraction

- **S**: **0** → dương, **1** → âm
- **Exponent**: phần số mũ
- **Fraction**: phần phân số
- **Bias**: đối với dạng chính xác đơn 32 bit thì có giá trị là **127**

Lưu ý: bởi vì giá trị của phần phân số luôn nhỏ hơn 1 nên khi chuyển giá trị phần này (đang ở dạng nhị phân) sang giá trị trong hệ thập phân thì trọng số của các bit sẽ lần lượt là -1,-2,… từ trái sang phải. Hay nói cách khác giá trị phần phân số sẽ được tính theo công thức: $Giá\ trị\ phân\ số = \sum_{i=-1}^{-23} x_i + 2^i$

Công thức tính:

Giá trị hệ thập phân = $(-1)^S \times (1 + \text{phân số}) \times 2^{\text{số mũ - bias}}$

Lưu ý: các đại lượng trong công thức đều đang xét ở hệ thập phân

Giá trị hệ nhị phân = $(-1)^S \times 1.\text{phân số} \times 10^{\text{số mũ - bias}}$

Lưu ý: đại lượng phân số trong công thức đang xét ở hệ nhị phân và kết quả thu được ta phải đổi sang hệ thập phân mới được kết quả cuối cùng

Ví dụ minh họa 1: Tính giá trị ở hệ thập phân của số thực dấu chấm động **0x41640000** theo chuẩn IEEE 754

- Bước 1: ta phải chuyển được giá trị từ hệ cơ số của đề bài cho sang giá trị ở hệ nhị phân (cụ thể ở đây đề bài cho giá trị ở dạng thập lục), giá trị sau khi đổi sang hệ nhị phân là: 0100 0001 0110 0100 0000 0000 0000 0000
  (4 1 6 4 0 0 0 0)
- Bước 2: giá trị nhị phân ở trên có độ dài 32 bit nên ta sẽ xác định các thành phần theo dạng chính xác đơn 32 bit:

1 bit	8 bits	23 bits
0	10000010	11001000000000000000000

  - Phần dấu S = 0 vì vậy đây là một số dương
  - Phần mũ E = $10000010_2 = 130_{10}$
  - Phần phân số F = $11001000000000000000000_2 = 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-5} = 0.78125_{10}$
  - Vì đây là số thực dấu chấm động chính xác đơn nên bias = 127_{10}
- Bước 3: áp dụng công thức để tính
  - Cách 1: áp dụng công thức tính giá trị thập phân

    Giá trị hệ thập phân = $(-1)^S \times (1 + \text{phân số}) \times 2^{\text{số mũ - bias}}$
    $= (-1)^0 \times (1 + 0.78125) \times 2^{130 - 127}$
    $= 14.25_{10}$
  - Cách 2: áp dụng công thức tính giá trị nhị phân

    Giá trị hệ nhị phân = $(-1)^S \times 1.\text{phân số} \times 10^{\text{số mũ - bias}}$
    $= (-1)^0 \times (1.11001)_2 \times 10^{130 - 127}$
    $= (-1)^0 \times (1.11001)_2 \times 10^3$
    $= (-1)^0 \times 1110.01_2$
    $= 1110.01_2 = 14.15_{10}$

Ví dụ minh họa 2: Biểu diễn giá trị 14.25_{10} theo dạng chuẩn IEEE 754 chính xác đơn

- Bước 1 (Xác định phần dấu S): vì $14.25_{10} > 0$ nên S = 0
- Bước 2 (Xác định phần số mũ E): ta đổi 14.25_{10} sang hệ nhị phân sẽ là 1110.01_2, bởi vì giá trị 1110.01_2 chưa ở dạng chuẩn ký hiệu khoa học nên ta sẽ chuyển nó về dạng chuẩn tắc (1.phân số) là $1.11001_2 \times 10^3$ (vì từ 1110.01_2 chuyển sang 1.11001_2 ta dịch chuyển dấu chấm về phía bên trái 3 bit nên sẽ $\times 10^3$). Ta có E = bias + 3 = 127 + 3 = $130_{10} = 10000010_2$
- Bước 3 (Xác định phần phân số F): F = $11001000000000000000000_2$
- Bước 4: ráp các phần tìm được ở trên theo chuẩn IEEE 754 ta được kết quả $01000001011001000000000000000000_2$ = $0x41640000_{16}$

CHƯƠNG 4: BỘ XỬ LÝ

➢ Các giai đoạn của mô hình thực thi lệnh

- Giai đoạn đọc lệnh (instruction fetch)
- Giai đoạn giải mã lệnh (instruction decode)
- Giai đoạn thực thi lệnh (instruction excute)
- Giai đoạn truy xuất bộ nhớ (memory access)
- Giai đoạn cập nhật kết quả (write back)

➢ Các nhóm lệnh và chức năng tương ứng cho từng nhóm lệnh của khối ALU (arithmetic logic unit)

- **Nhóm lệnh số học**: ALU có chức năng tính kết quả của phép toán cần thực hiện
- **Nhóm lệnh truy xuất dữ liệu**: ALU có chức năng xác định địa chỉ của bộ nhớ (load/store)
- **Nhóm lệnh rẻ nhánh**: ALU có chức năng xác định địa rẻ nhánh

➢ Đặc điểm quá trình thực thi lệnh

- Tất cả các lệnh khi được thực thi thì đều có 2 giai đoạn đầu là đọc lệnh và giải mã lệnh, sau 2 giai đoạn này thì tùy vào các nhóm lệnh khác nhau (nhóm lệnh số học, nhóm lệnh truy xuất dữ liệu, nhóm lệnh rẻ nhánh) sẽ yêu cầu các giai đoạn, tác vụ và khối chức năng khác nhau.
- Tất cả các lệnh trong cùng một nhóm lệnh sẽ có các tác vụ và các khối chức năng tham gia vào tính toán khá tương tự nhau
- Đối với lệnh số học, sau khi hai giá trị chứa trong hai thanh ghi nguồn đối với lệnh có sự tham gia của 2 toán hạng thanh ghi (hoặc 1 toán hạng thanh ghi và 1 toán hàng số nguyên) được truy xuất thì hai giá trị này được đưa vào khối luận lý số học (arithmetic logic unit – ALU) để thực hiện phép tính tương ứng (ví dụ phép cộng với lệnh add) đây được gọi là **giai đoạn thực thi lệnh (instruction excute)**
- Đối với lệnh truy xuất dữ liệu nội dung thanh ghi địa chỉ nền được truy xuất ở **giai đoạn giải mã lệnh** trong khi giá trị độ dời sẽ được rút trích trực tiếp trong lệnh, hai giá trị này được đưa vào khối ALU để thực hiện phép toán cộng. Kết quả của phép toán cộng này chính là địa chỉ ô nhớ chứa dữ liệu sẽ được truy xuất. Địa chỉ này được cung cấp cho **bộ nhớ dữ liệu (data memory)** để tiến hành truy xuất bộ nhớ dữ liệu, giai đoạn được gọi là **giai đoạn truy xuất bộ nhớ (memory access)**
- Đối với các lệnh rẻ nhánh có điều kiện thì hai thanh ghi toán hạng nguồn sau khi được truy xuất sẽ được đưa vào khối ALU để thực hiện phép trừ, nếu kết quả phép trừ là 0 thì hai thanh ghi chứa hai giá trị bằng nhau và ngược lại. Dựa vào kết quả của phép trừ này cùng với điều kiện của lệnh rẻ nhánh bằng (beq) hay không bằng (bne) mà giá trị thanh ghi PC sẽ được cập nhật lại bằng giá trị địa chỉ đích hay giữ nguyên giá trị sau giai đoạn đọc lệnh

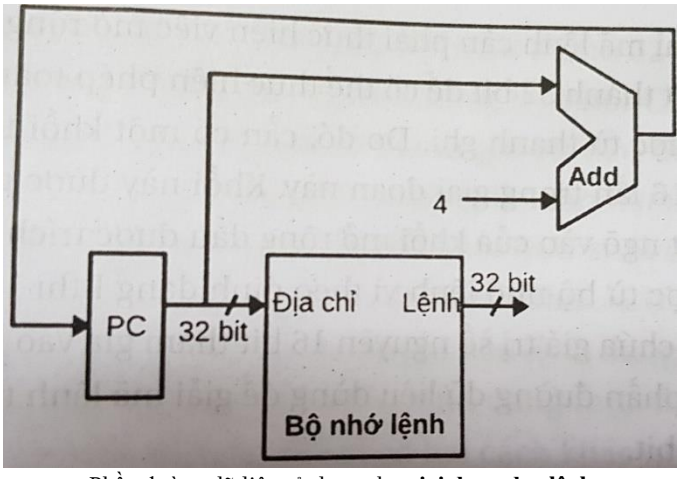
➢ Các thành phần cơ bản của một bộ xử lý

- Bộ nhớ lệnh (Instruction memory)
- Tập thanh ghi (Registers)
- Khối ALU (Arithmetic logic unit)
- Bộ nhớ dữ liệu (Data memory)

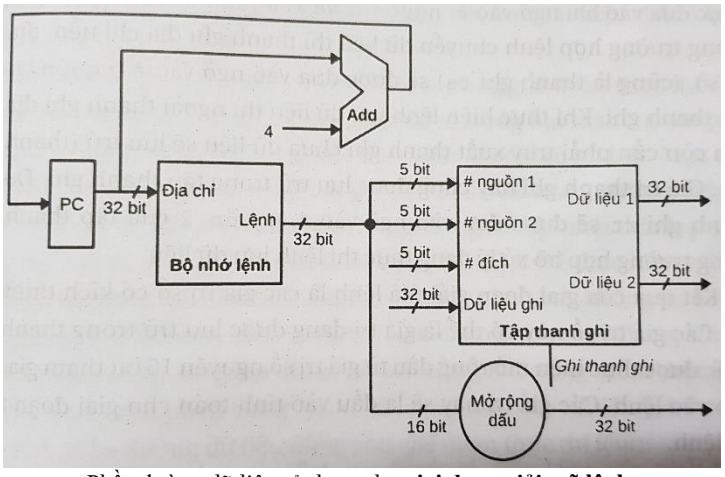
➢ Đường dữ liệu (Data path) và bộ điều khiển (Controller)

- Các thành phần của bộ xử lý sẽ hình thành nên một **đường dữ liệu (datapath)** của bộ xử lý, ngoài ra để quyết định xem các thành phần nào sẽ được tham gia vào hoạt động khi bộ xử lý đang thực thi một lệnh cụ thể thì cần có một khối chức năng có thể sinh ra các tín hiệu điều khiển sự hoạt động của các thành phần cơ bản trên khối chức năng này được gọi là **bộ điều khiển (controller)**
- Đường dữ liệu (Data path): gồm các **tín hiệu dữ liệu (data signal)** và các **khối chức năng (functional unit)** để xử lý các tín hiệu này. Nhiệm vụ chính của đường dữ liệu là xử lý dữ liệu dưới sự điều khiển của bộ điều khiển và truyền qua lại giữa các khối chức năng
- Bộ điều khiển (controller): chịu trách nhiệm phân tích lệnh đang được thực thi và sinh ra các tín hiệu để điều khiển *(ghi thanh ghi, đọc bộ nhớ, ghi bộ nhớ)* hoạt động của các khối chức năng

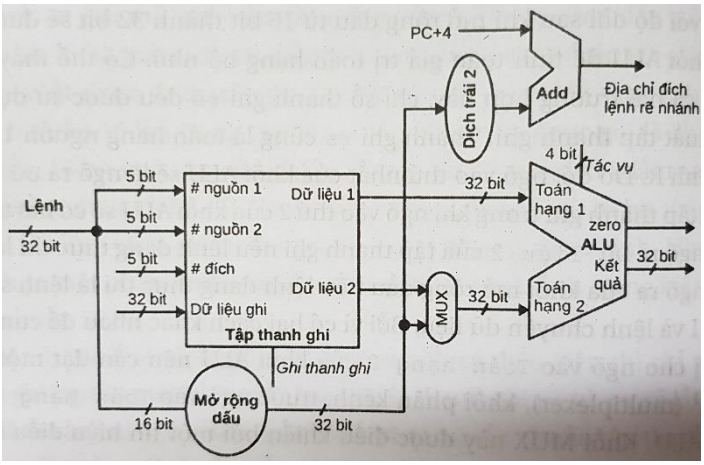
➢ Đường dữ liệu của các giai đoạn trong quá trình thực thi lệnh



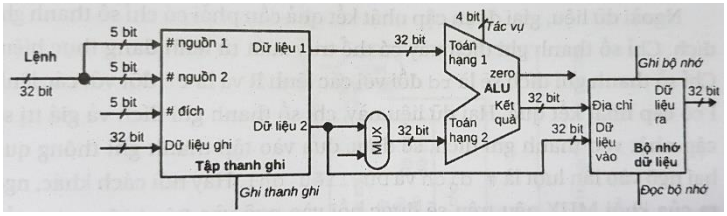
Phần đường dữ liệu sử dụng cho **giai đoạn đọc lệnh**



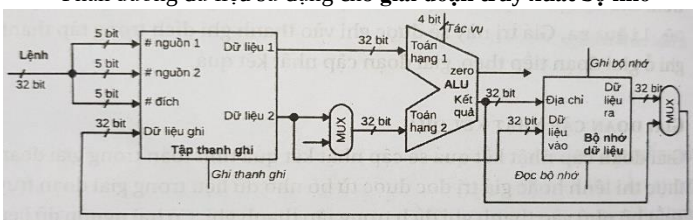
Phần đường dữ liệu sử dụng cho **giai đoạn giải mã lệnh**



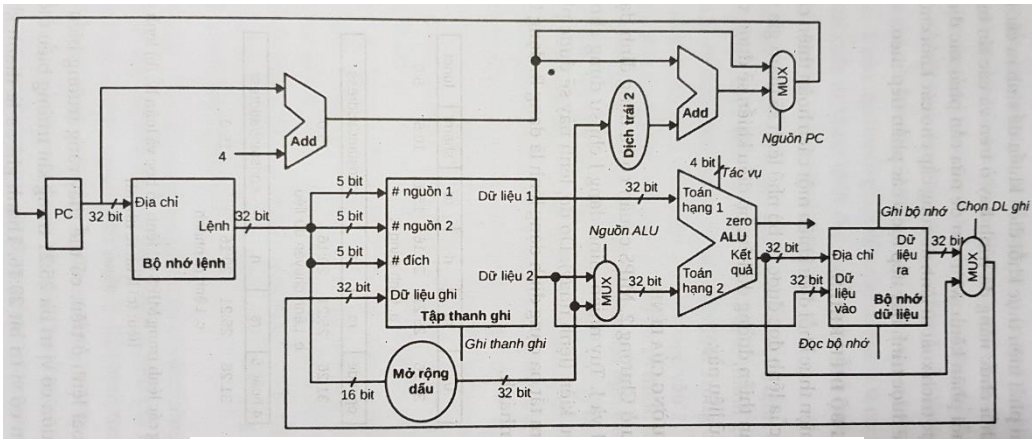
Phần đường dữ liệu sử dụng cho **giai đoạn thực thi lệnh**



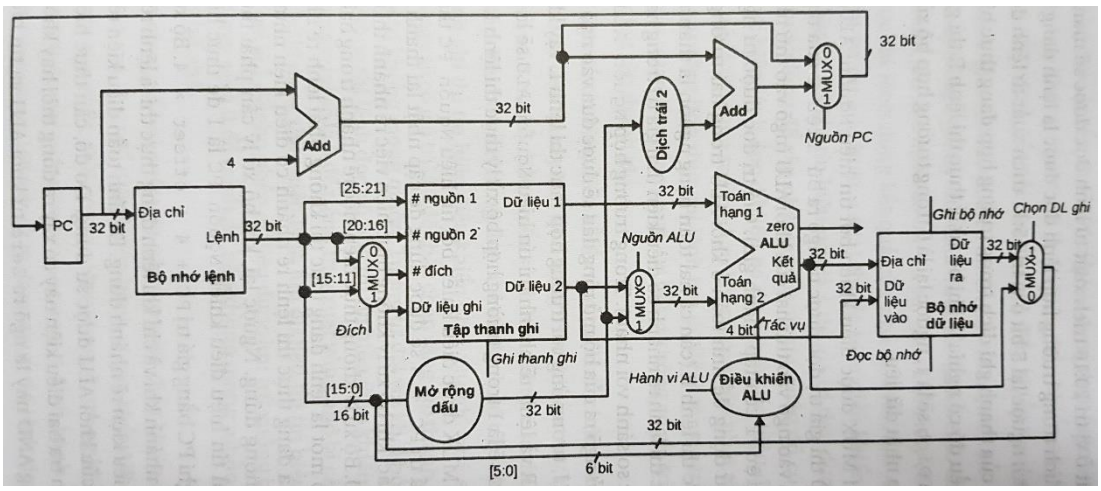
Phần đường dữ liệu sử dụng cho **giai đoạn truy xuất bộ nhớ**



Phần đường dữ liệu sử dụng cho **giai đoạn cập nhật kết quả**



Đường dữ liệu hoàn chỉnh theo mô hình đơn giản



Đường dữ liệu hoàn chỉnh theo mô hình đơn giản với đầy đủ các tín hiệu điều khiển và vị trí các bit trích xuất từ lệnh

➢ Giá trị của tín hiệu hành vi ALU và tác vụ tương ứng với các loại lệnh

Lệnh	Giá trị hành vi ALU (ALUOp)	Giá trị funct	Giá trị tác vụ (ALU control)	Tác vụ của ALU
lw	00	xxxxxx	0010	cộng
sw	00	xxxxxx	0010	cộng
beq / bne	01	xxxxxx	0110	trừ
and	10	100100	0000	and
or	10	100101	0001	or
add	10	100000	0010	cộng
sub	10	100010	0110	trừ
slt	10	101010	0111	so sánh
nor	10	100111	1100	nor

➢ Các ví dụ minh họa phân tích chi tiết các hoạt động của bộ xử lý MIPS

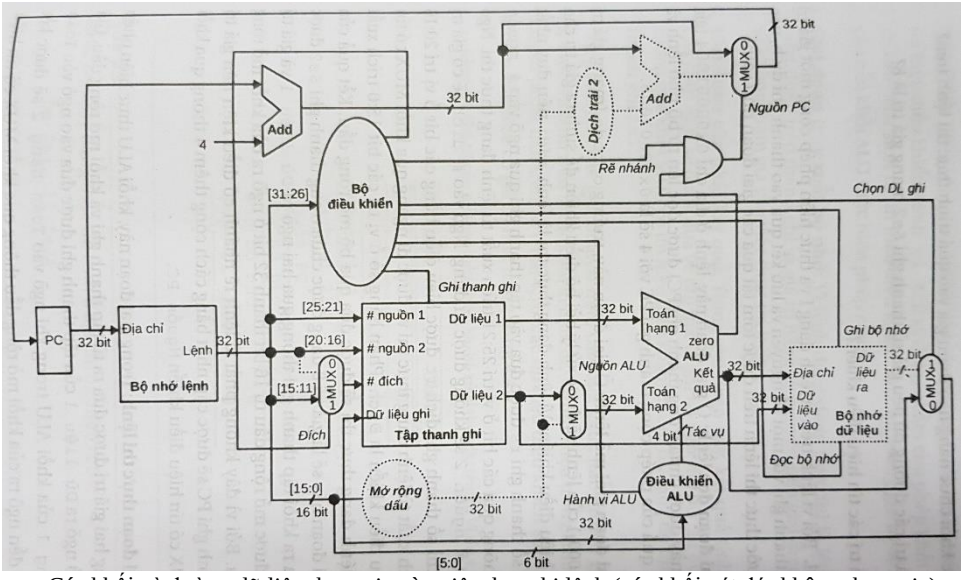
<u>Ví dụ 1:</u> giả sử bộ xử lý đang thực thi lệnh ở địa chỉ 0x00001004 : **add** $t0, $s1, $s2

- Bởi vì lệnh trên là lệnh dạng R và thực hiện phép cộng hai giá trị trong hai thanh ghi nguồn và lưu kết quả vào thanh ghi đích nên các bước thực thi lệnh gồm các giai đoạn sau (*chi tiết từng bước trong từng giai đoạn xem ở sách trang 163-164*):

(1) Giai đoạn đọc lệnh
(2) Giai đoạn giải mã lệnh
(3) Giai đoạn thực thi lệnh
(4) Giai đoạn cập nhật kết quả

Tín hiệu	Giá trị
Đích	1
Ghi thanh ghi	1
Nguồn ALU	0
Hành vi ALU	10
Nguồn PC	0
Ghi bộ nhớ	0
Đọc bộ nhớ	0
Chọn DL ghi	0

Bảng giá trị của tín hiệu điều khiển khi bộ xử lý thực hiện lệnh



Các khối và đường dữ liệu tham gia vào việc thực thi lệnh (các khối nét đứt không tham gia)

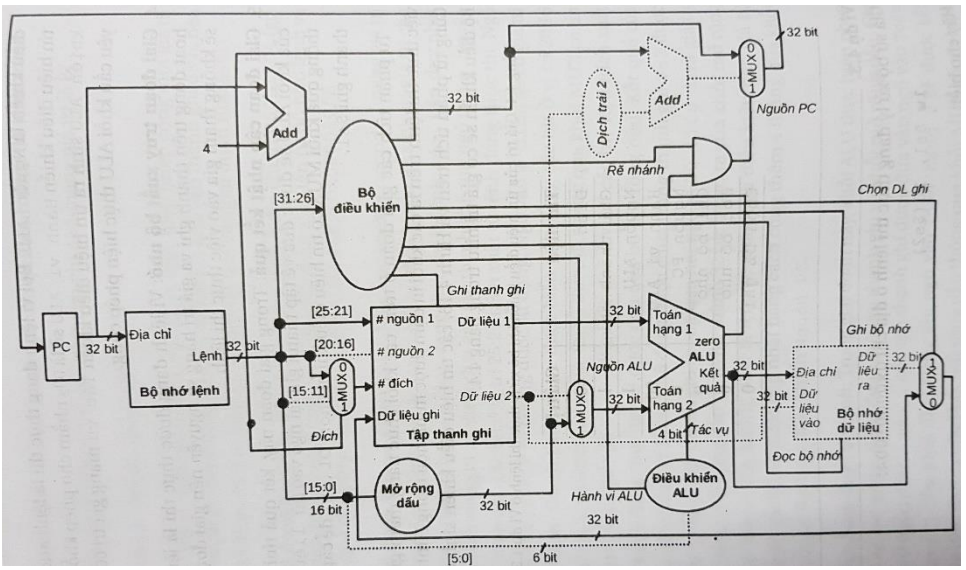
<u>Ví dụ 2:</u> giả sử bộ xử lý đang thực thi lệnh ở địa chỉ 0x00001004 : **addi** $s1, $s2, 10

- Bởi vì lệnh trên là lệnh dạng I và thực hiện phép cộng một giá trị trong thanh ghi nguồn với một số nguyên và lưu kết quả vào thanh ghi đích nên các bước thực thi lệnh gồm các giai đoạn sau (*chi tiết từng bước trong từng giai đoạn xem ở sách trang 166-167*):

(1) Giai đoạn đọc lệnh
(2) Giai đoạn giải mã lệnh
(3) Giai đoạn thực thi lệnh
(4) Giai đoạn cập nhật kết quả

Tín hiệu	Giá trị
Đích	0
Ghi thanh ghi	1
Nguồn ALU	1
Hành vi ALU	00
Nguồn PC	0
Ghi bộ nhớ	0
Đọc bộ nhớ	0
Chọn DL ghi	0

Bảng giá trị của tín hiệu điều khiển khi bộ xử lý thực hiện lệnh



Các khối và đường dữ liệu tham gia vào việc thực thi lệnh (các khối nét đứt không tham gia)

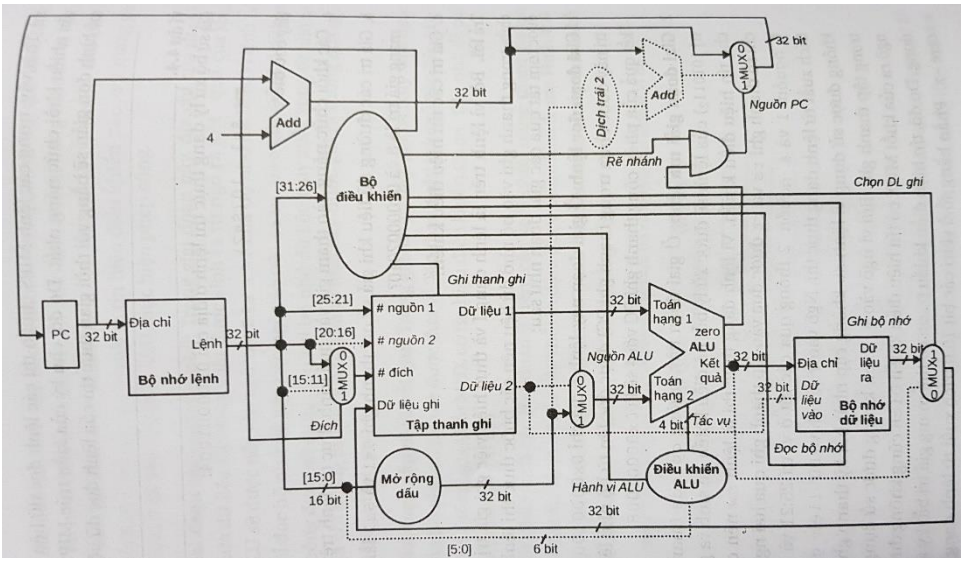
<u>Ví dụ 3:</u> giả sử bộ xử lý đang thực thi lệnh ở địa chỉ 0x00001004 : **lw** $s1, 10($s2)

- Bởi vì lệnh trên là lệnh dạng I và thực hiện việc đọc bộ nhớ dữ liệu và ghi kết quả đọc được vào thanh ghi đích nên các bước thực thi lệnh gồm các giai đoạn sau (*chi tiết từng bước trong từng giai đoạn xem ở sách trang 169-170*):

(1) Giai đoạn đọc lệnh
(2) Giai đoạn giải mã lệnh
(3) Giai đoạn thực thi lệnh
(4) Giai đoạn truy xuất bộ nhớ
(5) Giai đoạn cập nhật kết quả

Tín hiệu	Giá trị
Đích	0
Ghi thanh ghi	1
Nguồn ALU	1
Hành vi ALU	00
Nguồn PC	0
Ghi bộ nhớ	0
Đọc bộ nhớ	1
Chọn DL ghi	1

Bảng giá trị của tín hiệu điều khiển khi bộ xử lý thực hiện lệnh



Các khối và đường dữ liệu tham gia vào việc thực thi lệnh (các khối nét đứt không tham gia)

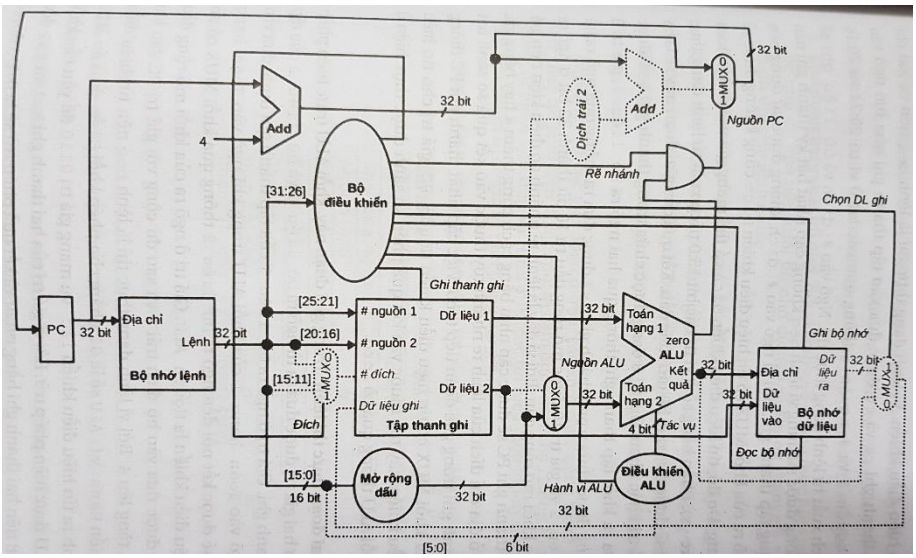
<u>Ví dụ 4:</u> giả sử bộ xử lý đang thực thi lệnh ở địa chỉ 0x00001004 : **sw** $s1, 10($s2)

- Bởi vì lệnh trên là lệnh dạng I và thực hiện việc ghi giá trị chứa trong thanh ghi nguồn vào bộ nhớ dữ liệu nên các bước thực thi lệnh gồm các giai đoạn sau (*chi tiết từng bước trong từng giai đoạn xem ở sách trang 171-172*):

(1) Giai đoạn đọc lệnh
(2) Giai đoạn giải mã lệnh
(3) Giai đoạn thực thi lệnh
(4) Giai đoạn truy xuất bộ nhớ

Tín hiệu	Giá trị
Đích	X (không có)
Ghi thanh ghi	0
Nguồn ALU	1
Hành vi ALU	00
Nguồn PC	0
Ghi bộ nhớ	1
Đọc bộ nhớ	0
Chọn DL ghi	X (không có)

Bảng giá trị của tín hiệu điều khiển khi bộ xử lý thực hiện lệnh



Các khối và đường dữ liệu tham gia vào việc thực thi lệnh (các khối nét đứt không tham gia)

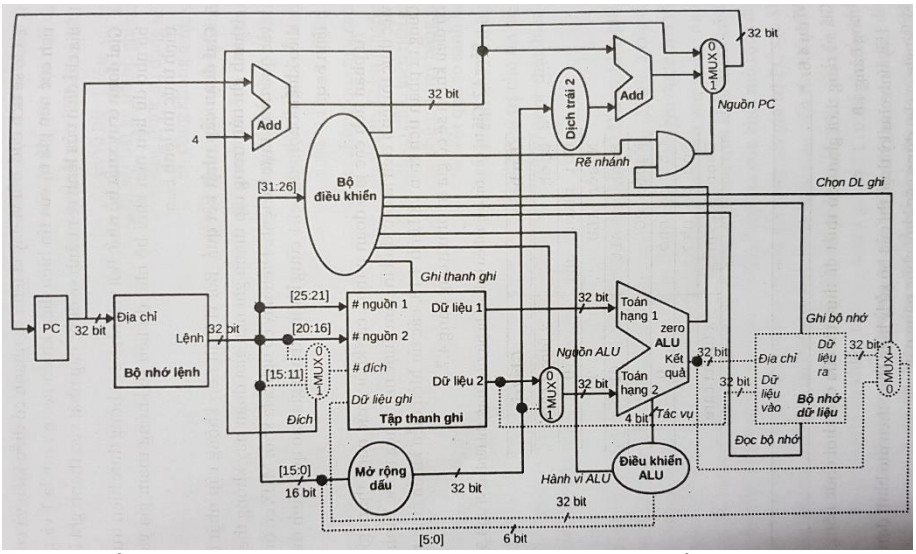
<u>Ví dụ 5:</u> giả sử bộ xử lý đang thực thi lệnh ở địa chỉ 0x00001004 : **beq** $s1, $s2, L1

- Bởi vì lệnh trên là lệnh rẻ nhánh có điều kiện nên các bước thực thi lệnh gồm các giai đoạn sau (*chi tiết từng bước trong từng giai đoạn xem ở sách trang 174,176*):

(1) Giai đoạn đọc lệnh
(2) Giai đoạn giải mã lệnh
(3) Giai đoạn thực thi lệnh

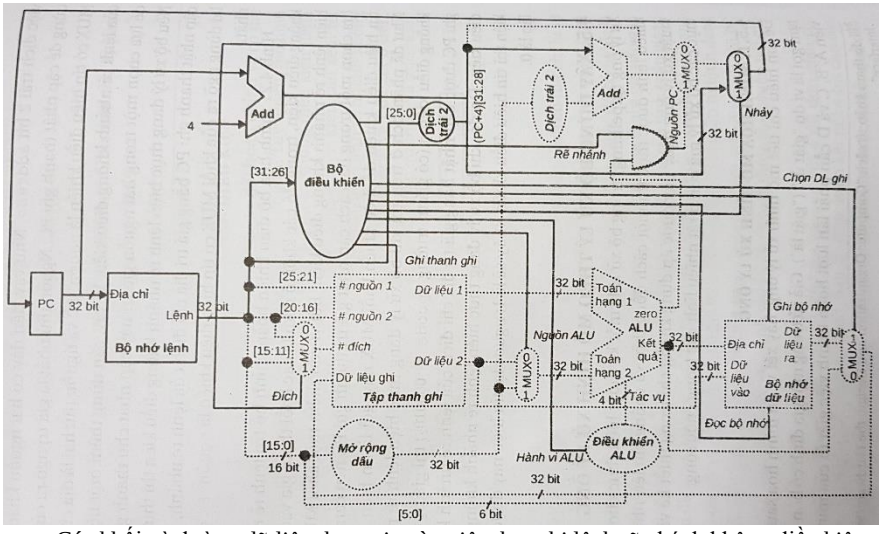
Tín hiệu	Giá trị
Đích	X (không có)
Ghi thanh ghi	0
Nguồn ALU	0
Hành vi ALU	01
Nguồn PC	1
Ghi bộ nhớ	0
Đọc bộ nhớ	0
Chọn DL ghi	X (không có)

Bảng giá trị của tín hiệu điều khiển khi bộ xử lý thực hiện lệnh



Các khối và đường dữ liệu tham gia vào việc thực thi lệnh (các khối nét đứt không tham gia)

➢ Bộ điều khiển hoàn chỉnh hỗ trợ lệnh rẻ nhánh không điều kiện j và jal



Các khối và đường dữ liệu tham gia vào việc thực thi lệnh rẽ nhánh không điều kiện
(các khối nét đứt không tham gia)