

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**



**Bài tập lớn:**  
**KIẾN TRÚC MÁY TÍNH**  
**(CO2008)**

**Case 01: Nhân Chia Hai Số Nguyên**

GVHD:	Trần Thanh Bình
Lớp - Nhóm:	DH_HK210 - L10
Sinh viên:	Trần Long Ân
MSSV:	1811486

## I. Giới thiệu

Trong lập trình máy tính, Hợp ngữ (hay assembly) thường được viết tắt là asm là bất kỳ ngôn ngữ lập trình cấp thấp nào có sự tương ứng rất mạnh giữa các tập lệnh trong ngôn ngữ và tập lệnh mã máy của kiến trúc. Bởi vì hợp ngữ phụ thuộc vào tập lệnh mã máy, mỗi trình biên dịch có hợp ngữ riêng được thiết kế cho chính xác một kiến trúc máy tính cụ thể. Hợp ngữ cũng có thể được gọi là mã máy tượng trưng (symbolic machine code).

Mã hợp ngữ được chuyển đổi thành mã máy thực thi bằng một chương trình được gọi là assembler. Quá trình chuyển đổi được gọi là assembling. Hợp ngữ thường có một câu lệnh trên một lệnh máy (1:1), nhưng các comment và các câu lệnh là chỉ thị trình biên dịch, macros, và các nhãn chương trình và địa chỉ bộ nhớ cũng được hỗ trợ.

Mỗi một hợp ngữ là dành riêng cho một kiến trúc máy tính cụ thể và đôi khi cho một hệ điều hành. Tuy nhiên, một số hợp ngữ không cung cấp cú pháp riêng cho lời gọi hệ điều hành, và hầu hết các hợp ngữ có thể được sử dụng phổ biến với bất kỳ hệ điều hành nào, vì ngôn ngữ này cung cấp quyền truy cập vào tất cả các khả năng thực sự của bộ xử lý, theo đó tất cả các cơ chế gọi hệ thống đều dừng lại. Trái ngược với hợp ngữ, hầu hết các ngôn ngữ lập trình bậc cao thường có khả năng di động trên nhiều kiến trúc nhưng yêu cầu thông dịch hoặc biên dịch, một công việc phức tạp hơn nhiều so với assembling.

Hợp ngữ đã từng được dùng rộng rãi trong tất cả các khía cạnh lập trình, nhưng ngày nay nó có xu hướng chỉ được dùng trong một số lãnh vực hẹp, chủ yếu để giao tiếp trực tiếp với phần cứng hoặc xử lý các vấn đề liên quan đến tốc độ cao điển hình như các trình điều khiển thiết bị, các hệ thống nhúng cấp thấp và các ứng dụng thời gian thực. Hợp ngữ MIPS

RICS- Reduced Instruction Set Computer, là một dạng của kiến trúc vi xử lý, là một phương pháp thiết kế các bộ vi xử lý theo hướng đơn giản hóa tập lệnh, trong đó thời gian thực thi tất cả các tập lệnh đều như nhau.

MIPS viết tắt của Microprocessor without Interlocked Pipeline Stages, là kiến trúc bộ tập lệnh RISC phát triển bởi MIPS Technologies. Ban đầu kiến trúc MIPS là 32bit, và

sau đó là phiên bản 64 bit. Nhiều sửa đổi của MIPS, bao gồm MIPS I, MIPS II, MIPS III, MIPS IV, MIPS V, MIPS32 và MIPS64. Phiên bản hiện tại là MIPS32 và MIPS64.

Kiến trúc MIPS là kiến trúc các thanh ghi. Tất cả các phép toán số học và logic đều chỉ xoay quanh thanh ghi(hay các hằng số được lưu trữ như là một thành phần của lệnh). Kiến trúc MIPS cũng bao gồm một số các câu lệnh đơn giản dùng để đọc dữ liệu từ bộ nhớ vào thanh ghi và ghi dữ liệu từ thanh ghi sang bộ nhớ. Vì vậy kiến trúc MIPS được gọi là kiến trúc nạp/lưu trữ, trong kiến trúc nạp/lưu trữ chỉ có những câu lệnh có thể truy cập vào bộ nhớ là các câu lệnh nạp/lưu trữ còn các câu lệnh khác chỉ có thể truy cập tới thanh ghi.

## II. Đề tài báo cáo

### 1. Giới thiệu đề tài

- Nhân, chia 2 số nguyên :
  - Cho 2 số nguyên (integers) có dấu A và B. Sử dụng hợp ngữ assembly MIPS, viết thủ tục nhân, chia 2 số nguyên A, B. Phép chia ra kết quả chia làm 2 phần, phần thương (bit cao) và phần dư (bit thấp). Chương trình hỗ trợ nhập vào số HEX hoặc số thập phân, kết quả xuất ra tương ứng với mode đã nhập.
  - Chú ý: không dùng trực tiếp phép nhân/chia, mà phải hiện thực giải thuật nhân/chia theo textbook/slide.

### 2. Ý tưởng

#### a) Chuyển đổi đầu vào 1 chuỗi ký tự sang hệ thập lục phân

- Trước tiên ta khai báo khoảng trống space để lưu chuỗi nhập vào.
- Sau ta nhập chuỗi vào bằng syscall như bình thường.
- Sử dụng 1 thanh ghi để đọc ký tự đầu của chuỗi đó:
  - + nếu ký tự đó không phải ký tự thuộc số thập phân thì ta báo lỗi.
  - + còn lại nếu ký tự hợp lệ ta chuyển từ ký tự sang số thập phân bằng cách trừ chúng cho mã ASCII thích hợp.
  - + nếu ký tự nhập vào là ký tự '\0' hoặc '\n' thì ta kết thúc đọc.
- Trong quá trình đọc ta sử dụng một biến đếm để đếm ký tự được nhập vào: nếu bên đếm bằng 8 'đã nhập được 8 ký tự' mà ký tự tiếp theo không phải '\n' hoặc '\0' thì ta báo lỗi.

- Khi đọc kí tự đầu tiên kiểm tra thử nếu đó là kí tự '-' - tức số âm, ta lưu vào một biến để đánh dấu và bỏ qua kí tự đầu, bắt đầu đọc từ kí tự thứ 2, đồng thời giảm kí tự biến đếm đi 1 (vì chuỗi nhập vào có dấu trừ nên ta mặc định tối đa 9 kí tự).

## b) Nhân 2 số

Dùng ý tương như nhân dưới dạng nhị phân : ví dụ trường hợp 0101 x 0010

```
  0 1 0 1
x 0 0 1 0
-----
  0 0 0 0
 0 1 0 1
 0 0 0 0
 0 0 0 0
= 0 0 0 1 0 1 0
```

Gọi các thanh ghi tương ứng trong phép nhân là A và B, biến lưu kết quả là C, ta thực hiện A x B:

- Trước tiên ta kiểm tra bit cuối cùng của B bằng cách dịch B sang trái 31 bit, và lưu vào 1 biến tạm để kiểm tra.
- Nếu biến tạm khác với 0 ta cộng C thêm cho A, nếu bằng 0 thì qua bỏ qua.
- Ta dịch A sang trái 1 bit (để khi cộng vào kết quả ta thụt vào 1 bit như trên phép tính ví dụ), dịch B sang phải 1 bit (để bỏ qua bit cuối đã đọc và tiếp tục đọc bit tiếp theo).
- Kiểm tra xem nếu A hoặc B bằng 0 (dịch hết các bit có giá trị 1) thì ta dừng vòng lặp, nếu không ta quay lại tiếp tục thực hiện đọc bit cuối của B.

### c) Chia 2 số

Phép chia tay : ví dụ 1011001 chia cho 110

```

      1 0 1 1 0 0 1
    -   1 1 0
    = 0 1 0 1 0 0 1      => 1
    -   1 1 0
    =  0 1 0 0 0 1      => 11
    -   1 1 0
    =   0 0 1 0 1      => 111
    -   1 1 0
    < 0 => dư 101      => 1110
  
```

Kết quả : phần nguyên : 1110, phần dư: 101

Ta thực hiện phép chia thanh ghi A cho B, C để lưu phần nguyên, như sau:

- Đầu tiên ta kiểm tra giá trị của B, nếu  $B = 0$  ta báo lỗi.
- Nếu B khác 0 ta lưu một biến nhớ bằng giá trị của B rồi dịch B sang phải cho đến khi  $A - B < 0$  (khi vào vòng lặp sẽ không bị ảnh hưởng và sẽ dịch lại sang trái 1 lần)
- Ta trừ A cho B, nếu  $A - B \geq 0$  thì ta cộng C cho 1 và cập nhật  $A = A - B$ , nếu khác thì ta bỏ qua.
- Dịch B sang phải 1 bit để tiếp tục cộng, kiểm tra nếu  $B < \text{biến nhớ ban đầu}$  (giá trị B lúc đầu chưa dịch) thì ta dừng vòng lặp. Nếu khác ta dịch C sang trái 1 bit để tiếp tục cộng.
- Kết thúc vòng lặp ta lưu giá trị của A còn lại cho phần dư và trả về kết quả.

### d) Kiểm tra số âm

- Để dễ dàng thực hiện các phép tính toán nhân chia, trước khi nhân và chia ta chuyển các số thành số dương.
- Sử dụng 1 biến đếm để lưu dấu, giả sử phép tính là  $A \times B$  hoặc  $A / B$ , nếu  $A < 0$  ta cộng biến đếm cho 1 rồi chuyển  $A = -A$ , nếu  $B < 0$  ta tiếp tục cộng cho 1 tương tự chuyển B thành -B.
- Sau khi thực hiện các phép nhân (chia) cho A và B, trước xuất kết quả ra màn hình ta kiểm tra biến đếm, nếu biến đếm = 1 ( $A < 0 \ \&\& \ B > 0$  hoặc  $A > 0 \ \&\& \ B < 0$ ) ta đổi dấu của kết quả, còn lại ta giữ nguyên (A B đều âm hoặc đều dương).

**e) Xuất kết quả phép là số thập lục phân**

- Syscall để in ra số nguyên thập phân thì ta gán \$v0 cho 1, còn khi là số thập lục phân thì là 34.
- Để nhanh chóng trong quá trình in ra kết quả, khi ta chọn phép tính cho hệ thập lục phân thì sử dụng 1 biến nhớ lưu giá trị là 33.
- Mỗi khi xuất kết quả ta cộng \$v0 cho biến này (tự động nếu là hệ thập lục phân sẽ chuyển sang 34 và in ra, còn nếu thập phân thì là \$v0 + 0 nên vẫn không thay đổi kết quả)

**3. Hoàn thiện dưới dạng Function cho các chức năng****a) Chức năng chuyển đổi giữa hệ thập phân sang hệ nhị phân**

Input : \$a0 = địa chỉ chuỗi

Output : \$v0 : thanh ghi mang giá trị đã được dịch

convert:

```
lb      $t0, 0($a0)
addi    $t1, $0, 0
addi    $t3, $0, 0
addi    $t4, $0, 0
bne     $t0, 45, loopConvert
addi    $a0, $a0, 1
lb      $t0, 0($a0)
addi    $t3, $t3, -1
addi    $t4, $t4, 1
```

loopConvert:

```
addi    $at, $0, 'f'
slt     $at, $at, $t0
bne     $at, $0, errorInput
slti    $at, $t0, '0'
bne     $at, $0, errorInput
addi    $at, $t0, 0xffffffff
slti    $at, $at, '9'
bne     $at, $0, number
slti    $at, $t0, 'a'
```

```
        beq        $at, $0, word
        j          errorInput
number:
        addi       $t0, $t0, -48
        j          checked
word:
        addi       $t0, $t0, -87
checked:
        add        $t1, $t1, $t0
        addi       $a0, $a0, 1
        lb        $t0, 0($a0)
        addi       $t3, $t3, 1
        beq        $t0, '\n', exit_convert
        beq        $t0, '\0', exit_convert
        beq        $t3, 8, errorSizeInput
        sll        $t1, $t1, 4
        j          loopConvert
exit_convert:
        addi       $v0, $t1, 0
        bne       $t4, 1, nextConvert
        sub        $v0, $0, $v0
nextConvert:
        jr         $ra
errorInput:
        la         $a0, error_input
        li         $v0, 4
        syscall
        li         $v0, 10
        syscall
errorSizeInput:
        la         $a0, error_size_input
        li         $v0, 4
        syscall
        li         $v0, 10
        syscall
```

## b) Chức năng nhân

Input : \$a0 = A , \$a1 = B

Output : \$v0 = A \* B

```
multiple:
    addi    $v0, $0, 0
loopMul:
    sll     $t0, $a1, 31
    beqz    $t0, nextMul
    add     $v0, $v0, $a0
nextMul:
    sll     $a0, $a0, 1
    srl     $a1, $a1, 1
    beqz    $a0, exitMul
    beqz    $a1, exitMul
    j       loopMul
exitMul:
    jr      $ra
```

## c) Chức năng chia số thập phân

Input : \$a0 = A, \$a1 = B

Output : \$v0 = A / B , \$v1 = A % B

```
divide:
    addi    $t4, $a1, 0
    addi    $v0, $0, 0
loop_div1:
    sub     $t2, $a0, $a1
    bltz    $t2, loop_div2
    sll     $a1, $a1, 1
    j       loop_div1
loop_div2:
    sub     $t1, $a0, $a1
    bltz    $t1, next_divv
    addi    $v0, $v0, 1
```



```
        addi    $a0, $t1, 0
next_divv:
        srl     $a1, $a1, 1
        slt     $at, $a1, $t4
        bne     $at, $0, exit_div
        sll     $v0, $v0, 1
        j       loop_div2
exit_div:
        addi    $v1, $a0, 0
        jr      $ra
```

### III. Testcases

```
Select the type of number entered : (1) - The decimal number (2) - The hexadecimal number : 1

Enter A : 12424

Enter B : 1233

Choose the operation using : (1) - Multiplication (2) - Division : 1

Result : A * B = 15318792
-- program is finished running --
```

```
Select the type of number entered : (1) - The decimal number (2) - The hexadecimal number : 1

Enter A : -12312

Enter B : 2424

Choose the operation using : (1) - Multiplication (2) - Division : 1

Result : A * B = -29844288
-- program is finished running --
```

```
Select the type of number entered : (1) - The decimal number (2) - The hexadecimal number : 1

Enter A : -41224

Enter B : -1321

Choose the operation using : (1) - Multiplication (2) - Division : 1

Result : A * B = 54456904
-- program is finished running --
```

```
Select the type of number entered : (1) - The decimal number (2) - The hexadecimal number : 1

Enter A : 42143

Enter B : 1233

Choose the operation using : (1) - Multiplication (2) - Division : 2

Result : A / B = 34
Result : A % B = 221
-- program is finished running --
```

```
Select the type of number entered : (1) - The decimal number (2) - The hexadecimal number : 1

Enter A : -463635

Enter B : 2424

Choose the operation using : (1) - Multiplication (2) - Division : 2

Result : A / B = -191
Result : A % B = -651
-- program is finished running --
```

```
Select the type of number entered : (1) - The decimal number (2) - The hexadecimal number : 1

Enter A : 12424

Enter B : 0

Choose the operation using : (1) - Multiplication (2) - Division : 2

Error !!! Divisor must be non-zero!
-- program is finished running --
```

```
Select the type of number entered : (1) - The decimal number (2) - The hexadecimal number : 1

Enter A : -21352

Enter B : -1242

Choose the operation using : (1) - Multiplication (2) - Division : 2

Result : A / B = 17
Result : A % B = 238
-- program is finished running --
```

```
Select the type of number entered : (1) - The decimal number (2) - The hexadecimal number : 2

Enter A : 124ab46

Enter B : ac34

Choose the operation using : (1) - Multiplication (2) - Division : 1

Result : A * B = 0xde85d238
-- program is finished running --
```

```
Select the type of number entered : (1) - The decimal number (2) - The hexadecimal number : 2

Enter A : -24a

Enter B : 23e

Choose the operation using : (1) - Multiplication (2) - Division : 1

Result : A * B = 0xffffade14
-- program is finished running --
```

```
Select the type of number entered : (1) - The decimal number (2) - The hexadecimal number : 2

Enter A : -ba12423

Enter B : -241ed

Choose the operation using : (1) - Multiplication (2) - Division : 2

Result : A / B = 0x00000526
Result : A % B = 0x0001b9f5
-- program is finished running --
```

```
Select the type of number entered : (1) - The decimal number (2) - The hexadecimal number : 2

Enter A : 121225254
The number entered must be less than or equal to 8 bits!
-- program is finished running --
```

```
Select the type of number entered : (1) - The decimal number (2) - The hexadecimal number : 2

Enter A : agqgegq

The number entered must be hexadecimal!
-- program is finished running --

Select the type of number entered : (1) - The decimal number (2) - The hexadecimal number : 5

Only two options are 1 or 2!
Select the type of number entered : (1) - The decimal number (2) - The hexadecimal number : 2

Enter A : 64ab35

Enter B : -532

Choose the operation using : (1) - Multiplication (2) - Division : 5

Only two options are 1 or 2!
Choose the operation using : (1) - Multiplication (2) - Division : 2

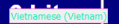
Result : A / B = 0xffffeca0
Result : A % B = 0xfffffd8b
-- program is finished running --
```

## IV. Thông kê lệnh

Các lệnh được sử dụng ở bài báo cáo :

- R-format Instructions: add, sub, sll, srl, jr
- I-format Instructions: beq, bne, addi, bgez, lb, bnez, beqz, bltz, lui, ori
- J-format Instructions: j, jar

Ý nghĩa của các lệnh:

Name	Format	Layout						Example
			5 bits	5 bits	5 bits	5 bits	6 bits	
		op	rs	rt	rd	shamt	funct	
add	R	0	2	3	1	0	32	add \$1, \$2, \$3
addu	R	0	2	3	1	0	33	addu \$1, \$2, \$3
sub	R	0	2	3	1	0	34	sub \$1, \$2, \$3
subu	R	0	2	3	1	0	35	subu \$1, \$2, \$3
and	R	0	2	3	1	0	36	and \$1, \$2, \$3
or	R	0	2	3	1	0	37	or \$1, \$2, \$3
nor	R	0	2	3	1	0	39	nor \$1, \$2, \$3
slt	R	0	2	3	1	0	42	slt \$1, \$2, \$3
sltu	R	0	2	3	1	0	43	sltu \$1, \$2, \$3
sll	R	0	0	2	1	10	0	sll \$1, \$2, 10
srl	R	0	0	2	1	10	2	srl \$1, \$2, 10
jr	R	0	31	0	0	0	8	jr \$31

Hình 1- R-format instruction (Nguồn : <http://www.cs.kzoo.edu/>)

Name	Format	Layout						Example
		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	
		op	rs	rt	immediate			
beq	I	4	1	2	25 (offset)			beq \$1, \$2, 100
bne	I	5	1	2	25 (offset)			bne \$1, \$2, 100
addi	I	8	2	1	100			addi \$1, \$2, 100
addiu	I	9	2	1	100			addiu \$1, \$2, 100
andi	I	12	2	1	100			andi \$1, \$2, 100
ori	I	13	2	1	100			ori \$1, \$2, 100
slti	I	10	2	1	100			slti \$1, \$2, 100
sltiu	I	11	2	1	100			sltiu \$1, \$2, 100
lui	I	15	0	1	100			lui \$1, 100
lw	I	35	2	1	100 (offset)			lw \$1, 100(\$2)
sw	I	43	2	1	100 (offset)			sw \$1, 100(\$2)

Hình 2- I-format Instructions (Nguồn : <http://www.cs.kzoo.edu/>)

Name	Format	Layout						Example
		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	
		op	address					
j	J	2	2500					j 10000
jal	J	3	2500					jal 10000

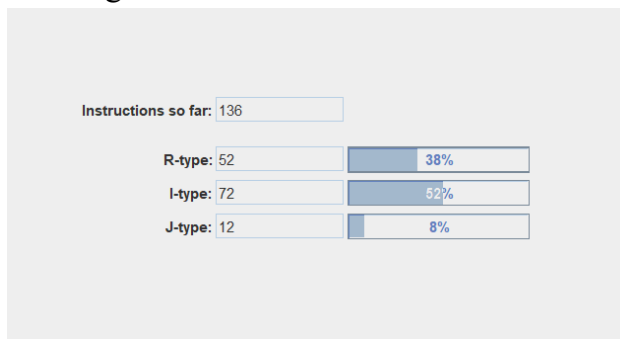
Hình 3- J-format Instructions (Nguồn : <http://www.cs.kzoo.edu/>)

## V. Tính toán thời gian chạy

- Công thức :  $\text{Time} = (\text{Instruction Count} * \text{CPI}) / \text{Clock Rate}$
- Trong đó :  
 Time : thời gian thực thi chương trình  
 Instruction Count : tổng số lệnh  
 CPI : số chu kì xung nhịp của 1 lệnh (được cho bằng 1)  
 Clock Rate: tần số (được cho bằng 2GHz)
- Số lệnh và loại lệnh khi tính phụ thuộc vào các tham số đầu vào nên ta sẽ tính toán trên các trường hợp ví dụ cụ thể:

- Ví dụ 1:  $121224 * (-1232) = -149347968$

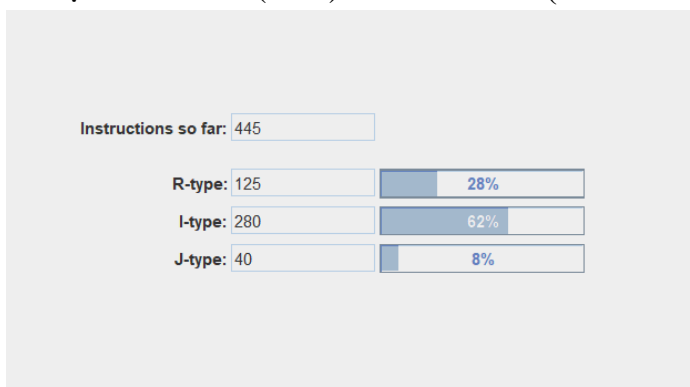
Sử dụng tool đếm các lệnh:



Thay vào công thức :

$$\Rightarrow \text{Time} = (136 * 1) / (2 * 10^9) = 6.8 * 10^{-8} (\text{s})$$

- Ví dụ 2:  $ab24a2 / (-c23) = 0xfffff1e7$  (dư  $0xfffff4c9$ )



Thay vào công thức :

$$\Rightarrow \text{Time} = (445 * 1) / (2 * 10^9) = 2.225 * 10^{-7} (\text{s})$$