Computer Architecture – LAB 3

김원표
kwp@hallym.ac.kr

Index

- Input from standard input
- Valid bits
- Multiplication
- Division

Lab 3

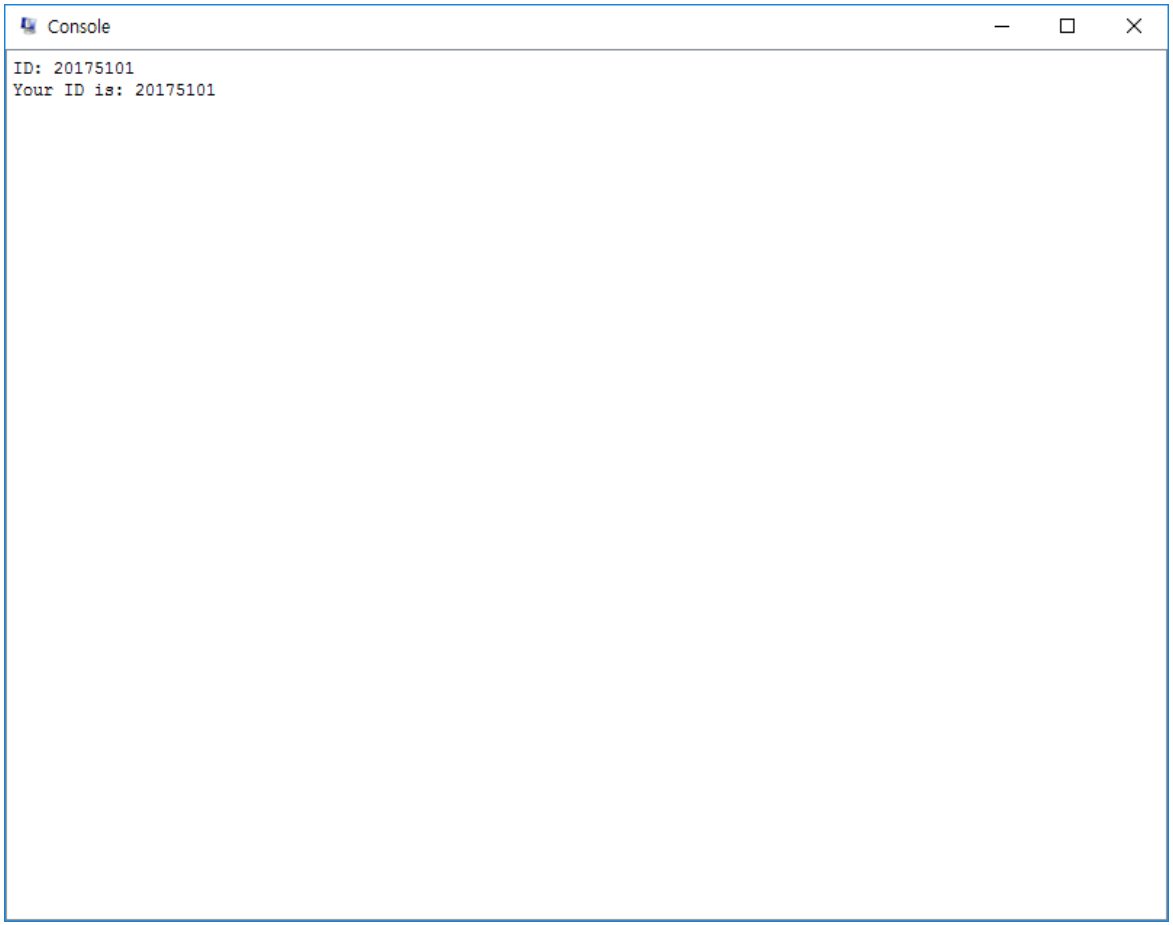
- lab3_1.asm – 실습 1

```
.data
std_id: .asciiz "ID: "
std_ret: .asciiz "Your ID is: "

.text
.globl main

main:
    # system code 4: print string
    # $a0 will point out 'std_id'
    li $v0, 4
    la $a0, std_id
    syscall

    # system code 5: read integer
    # $v0 will have the entered value
    li $v0, 5
    syscall
```

```
# Below three sentences denote the same action.
# $s1 == $v0 -- $s1 <- $v0 or $0
# $s2 == $v0 -- $s2 <- $v0
# $s3 == $v0 -- $s3 <- $v0 add $0
#
# These three sentences will transfer the figure in
# $v0 to other registers ($s1-$s3, respectively).
or $s1, $v0, $0
move $s2, $v0
add $s3, $v0, $0

# system code 4: print string
# $a0 will point out 'std_ret'
li $v0, 4
la $a0, std_ret
syscall

# system code 1: print integer value
# $a0 will have the same value as '$s1'
li $v0, 1
move $a0, $s1
syscall

# system code 10: exit program
li $v0, 10
syscall
```

Lab 3

- **lab3_1.asm – 과제 1**
  - 코드 주석 전체 해석
  - 결과를 확인하고 아래 결과가 나온 이유를 설명



Lab 3

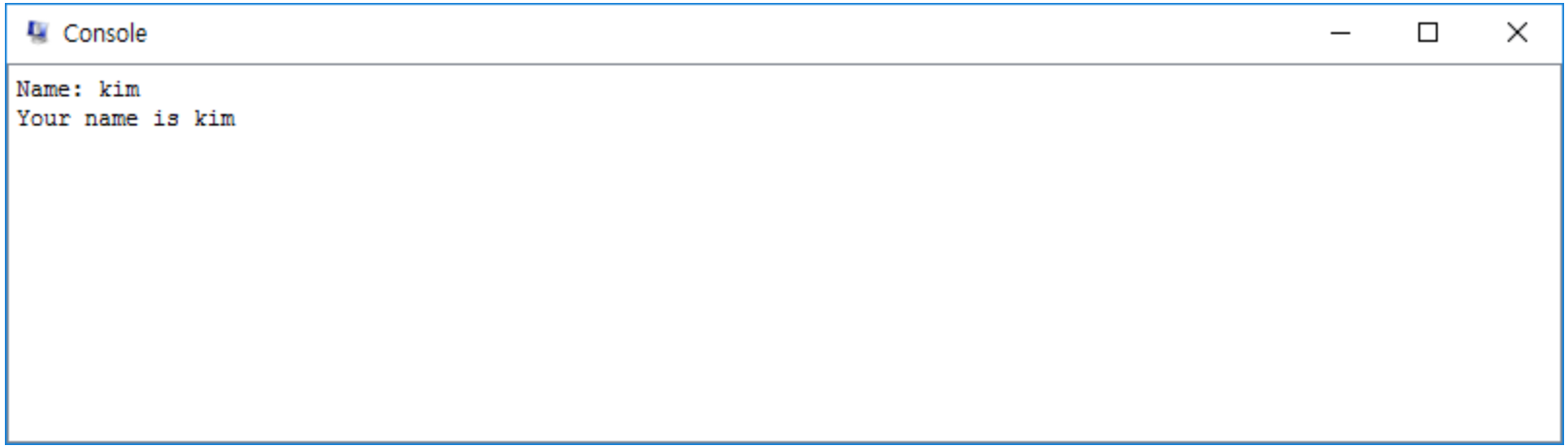
- lab3_2.asm – 실습 2

```
.data
std_ask:    .asciiz "Name: "
std_ret:    .asciiz "Your name is "
std_name:   .space 20    # empty buffer

.text
.globl main
main:
    # system code 4: print string
    # $a0 will point out 'std_id'
    li $v0, 4
    la $a0, std_ask
    syscall

    # $a0 will point out 'std_name'
    # We can input only 20-byte (19 ascii characters)
    la $a0, std_name
    li $a1, 20
```

```
# system code 8: read string
#
# $a0 : address where string to be sotred
# $a1 : length of string buffer
li $v0, 8
syscall

# system code 4: print string
# $a0 will point out 'std_ret'
li $v0, 4
la $a0, std_ret
syscall

# system code 4: print string
# $a0 will point out 'std_name'
li $v0, 4
la $a0, std_name
syscall

# system code 10: exit program
li $v0, 10
syscall
```

Lab 3

- **lab3_2.asm – 과제 2**
  - 코드 주석 전체 해석
  - 결과를 확인하고 아래 결과가 나온 이유를 설명



Lab 3

- **Valid bits**
  - It means the number of effective bits among a set of bits.
- **Positive numbers and unsigned numbers**
  - Start from the leftmost digit 1 until the rightmost digit.

The number of valid bits is 19							
0000	0000	0000	0101	0111	1011	1111	0000

- **Negative numbers**
  - Start from the left most digit 0 until rightmost digit.

The number of valid bits is 23							
1111	1111	1001	1011	1101	0001	1010	0010

Lab 3

- If you want to denote multiplication of two N-digit decimal numbers, then you need up to 2N-digit.
  - e.g) $9 \times 9 = 81$

- If you want to denote multiplication of two N-digit binary numbers, then you need up to 2N-bit.
  - e.g) $1111 \times 1111 = 1110\ 0001$

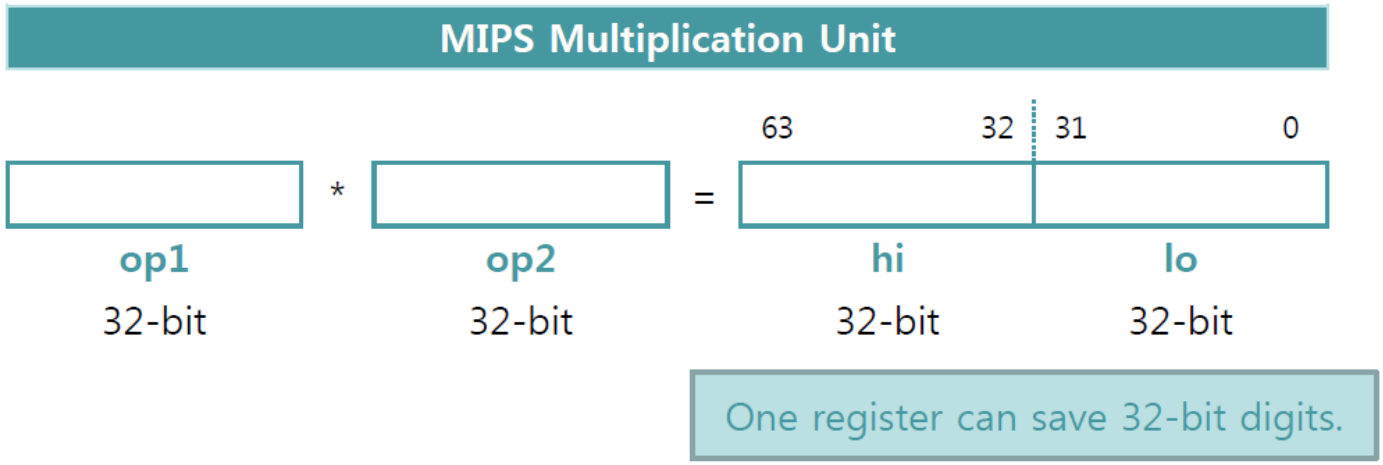
- MULT
  - mult $s1, $s2 # $s1 * $s2
  - Operands should be recognized signed numbers.

- MULTU
  - multu $s1, $s2 # $s1 * $s2
  - Operands should be recognized unsigned numbers.

Lab 3

- **MIPS Multiplication Unit**

- **MIPS Multiplication Unit stores the result in both HI and LO registers.**
  - Both registers are not general purpose registers.



Lab 3

- The Use of MFHI and MFLO
- MFHI and MFLO instructions will transfer the figures in HI and LO registers to regular registers.
  - MFHI $s1 # The figure in HI register will move to $s1.
  - MFLO $s2 # The figure in LO register will move to $s2.
- HI and LO registers can't be used for any logical or arithmetical instructions.
- If you want to use the result of multiplication, then you should move the figures in HI and LO registers to other general registers.

Lab 3

- lab3_3.asm – 실습 3

```
.text
.globl main

main:
    # The first calculation.. 0xABCDEF * 0xABCD
    # $t1 <- 0xABCDEF
    # $t2 <- $0 or 0xABCD
    la $t1, 0xABCDEF
    ori $t2, $0, 0xABCD

    # $t1 and $t2 will be multiplied.
    # The result will be stored in both hi and lo registers.
    # The figure in hi register will move to $s1.
    # The figure in lo register will move to $s2.
    mult $t1, $t2
    mfhi $s1
    mflo $s2

    # The second calculation.. 0xFFFFFFFF * 0xABCD
    # $t1 <- 0xFFFFFFFF
    # $t2 <- $0 or 0xABCD
    la $t1, 0xFFFFFFFF
    ori $t2, $0, 0xABCD
    # $t1 and $t2 will be multiplied.
    # The result will be stored in both hi and lo registers.
    # The figure in hi register will move to $s3.
    # The figure in lo register will move to $s4.
    mult $t1, $t2
    mfhi $s3
    mflo $s4

    # The final calculation.. 0xCDEF * 0xCDEF
    # $t1 <- 0xCDEF
    # $t2 <- $0 or 0xCDEF
    la $t1, 0xCDEF
    ori $t2, $0, 0xCDEF

    # $t1 and $t2 will be multiplied.
    # The result will be stored in both hi and lo registers.
    # The figure in lo register will move to $s5.
    # The figure in lo register will move to $s6.
    mult $t1, $t2
    mfhi $s5
    mflo $s6

    # system code 10: exit program
    li $v0, 10
    syscall
```

- Result

```
R17 [s1] = 73
R18 [s2] = 4c228d63
R19 [s3] = ffffffff
R20 [s4] = ffff5433
R21 [s5] = 0
R22 [s6] = a5a8a521
```

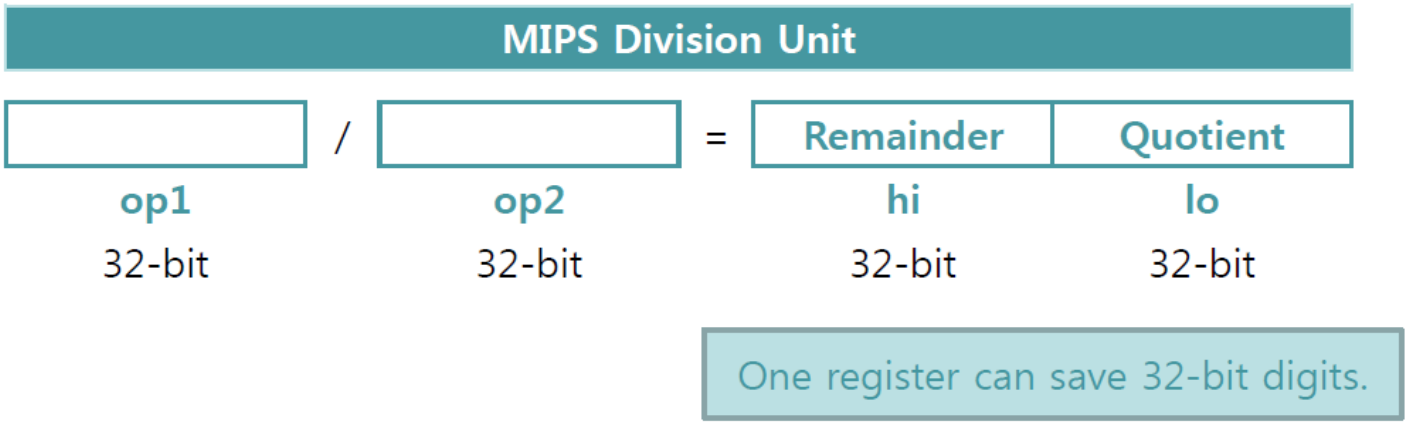
Lab 3

- DIV
  - div $s1, $s2 # $s1 / $s2
  - Operands should be recognized signed numbers.
- DIVU (U: Unchecked)
  - divu $s1, $s2 # $s1 / $s2
  - Operands should be recognized unsigned numbers.
- HI and LO registers
  - HI register : the remainder
  - LO register : the quotient

Lab 3

- **MIPS Division Unit**

- **MIPS Division Unit stores the result in both HI and LO registers.**
  - Both registers are not general purpose registers.



Lab 3

- The Use of MFHI and MFLO
- MFHI and MFLO instructions will transfer the figures in HI and LO registers to regular registers.
  - MFHI $s1 # The remainder (HI) will move to $s1 register.
  - MFLO $s2 # The quotient (LO) will move to $s2 register.
- HI and LO registers can't be used for any logical or arithmetical instructions.
- If you want to use the result of division, then you should move the figures in HI and LO registers to other general registers.

Lab 3

- lab3_4.asm – 실습 4

```
.text
.globl main

main:
    # The first calculation.. 811 / 10
    # $t1 <- 811
    # $t2 <- $0 or 10
    la $t1, 811
    ori $t2, $0, 10

    # $t1 (dividend) and $t2 (divisor) will be divided.
    # The result will be stored in both hi and lo registers.
    # The figure in hi (remainder) register will move to $s1.
    # The figure in lo (quotient) register will move to $s2.
    div $t1, $t2
    mfhi $s1
    mflo $s2

    # The second calculation.. -6 / 2
    # $t1 <- -6
    # $t2 <- $0 or 2
    la $t1, -6
    ori $t2, $0, 2

    # $t1 (dividend) and $t2 (divisor) will be divided.
    # The result will be stored in both hi and lo registers.
    # The figure in hi (remainder) register will move to $s3.
    # The figure in lo (quotient) register will move to $s4.
    div $t1, $t2
    mfhi $s3
    mflo $s4

    # The third calculation.. 100 / 100
    # $t1 <- 100
    # $t2 <- $0 or 100
    la $t1, 100
    ori $t2, $0, 100

    # $t1 (dividend) and $t2 (divisor) will be divided.
    # The result will be stored in both hi and lo registers.
    # The figure in hi (remainder) register will move to $s5.
    # The figure in lo (quotient) register will move to $s6.
    div $t1, $t2
    mfhi $s5
    mflo $s6

    # system code 10: exit program
    li $v0, 10
    syscall
```

- Result

```
R17 [s1] = 1
R18 [s2] = 51
R19 [s3] = 0
R20 [s4] = fffffffd
R21 [s5] = 0
R22 [s6] = 1
```

Lab 3

- 과제 3

- 1

  $s1 : 0xABCDEF

  $s2 : save 10 or lower number through console.

- 2

  Calculate $s1 MULT / DIV $s2.

  Then save these results into from $s3 to $s6.

  (MULT -> $s3(HI) and $s4(LO) DIV -> $s5(HI) and $s6(LO))

- **과제**
  - 실습한 내용의 화면캡쳐 본
  - **실습 코드(실습 1, 2, 3, 4), 과제 1, 2, 3, 전체 주석 해석**
  - 워드 문서로 합하여 제출

- **파일명 ex) ca_03_학번_이름.docx**
  - **스마트 캠퍼스 과제란 제출 – 파일명 엄수**

- **제출기한**
  - 10월 26일 23:59까지

- **수업시간 내 완료시 조교의 확인을 받고 퇴실 가능, 미확인시 결석처리**