

MIPS Assembly Programming

21511774

컴퓨터공학과

이성근

과제 1: 산술연산

```
.data
    A: .word 4
    B: .word 2
    C: .word 10
    D: .word 2
    value: .word 0
    msg: .asciiz "value: "
```

변수 선언부

```
.text
.globl main
```

main:

```
lw      $s0, A
lw      $s1, B
lw      $s2, C
lw      $s3, D
```

레지스터의 변수저장

```
add      $t0, $s0, $s1
sub      $t1, $s2, $s3
```

A+B

C-D

```
mult     $t0, $t1
mflo     $t0
```

\$t0 x \$t1

```
sw      $t0, value
```

value에 저장

```
li      $v0, 4
la      $a0, msg
syscall
```

```
li      $v0, 1
lw      $a0, value
syscall
```

```
li      $v0, 10
syscall
```

Terminal

value: 48

과제 3: 피보나치 수

추가 구현

☑ -1을 입력할 때까지 반복합니다

☑ 연산 값의 과정을 출력합니다.

.data

```
fibs: .word 0 : 20      입력 가능한 x의 최대값=20
size: .word             반복할 횟수
prompt: .asciiz "input X for fibonacci(x): "
prompt2: .asciiz "input X for fibonacci(x): "
result: .asciiz "result: "
newline: .asciiz "\n"
space: .asciiz " + "
equal: .asciiz " = "
```

.text

start:

```
la $a0, prompt
li $v0, 4      Input X for fibonacci(x)를
syscall        출력하고 입력받음
li $v0, 5
syscall
```

check:

```
li $s4, -1
beq $s4, $v0, end    입력값이 -1이라면 종료
```

init:

```
la $t0, fibs      선언한 Data를 load
la $t5, size
addi $v0, $v0, 1   피보나치 항이 0부터 시작하기 때문에
sw $v0, 0($t5)     입력값에 1을 더해주어 0부터 x까지 출력
lw $t5, 0($t5)
li $t2, 1
li $t3, 0
li $t6, 1          f(0) = 0
sw $t3, 0($t0)     f(1) = 1
sw $t2, 4($t0)     f(0)과 f(1)을 미리 선언하였으므로
addi $t1, $t5, -2   입력값에다가 -2를 더해주어 반복횟수 설정
```

loop:

```
lw $t3, 0($t0)
lw $t4, 4($t0)
add $t2, $t3, $t4
```

```

sw    $t2, 8($t0)
addi  $t0, $t0, 4
addi  $t1, $t1, -1
bgtz  $t1, loop
la    $a0, fibs
add   $a1, $zero, $t5
jal   print
li    $v0, 10
syscall

```

**f(2)부터 시작하여 값을 저장하고 주소값을
4만큼 이동함**

print:

```

add   $t0, $zero, $a0
add   $t1, $zero, $a1
la    $a0, result
li    $v0, 4
syscall

```

result: 를 출력

out:

```

lw    $a0, 0($t0)
li    $v0, 1
syscall
addi  $t0, $t0, 4
addi  $t1, $t1, -1
bgtz  $t1, add
la    $a0, newline
li    $v0, 4
syscall
la    $a0, prompt2
li    $v0, 4
syscall
j     start

```

**f(0)부터 f(x)까지 연산값의 과정과
결과를 출력**

eq:

```

la    $a0, equal
li    $v0, 4
syscall
j     out

```

마지막 항이라면 ' = '를 출력

add:

```

beq   $t1, $t6, eq
la    $a0, space
li    $v0, 4
syscall
j     out

```

**마지막 항인지 검사하여 마지막 항이라면
eq로 점프,
그렇지 않다면 ' + '를 출력**

end:

```

jr    $ra

```

시스템을 종료시킴

실행 결과

```
Terminal IRQ 0 ffff0000
value: 48input X for fibonacci(x): 5
result: 0 + 1 + 1 + 2 + 3 = 5
input X for fibonacci(x): 6
result: 0 + 1 + 1 + 2 + 3 + 5 = 8
input X for fibonacci(x): 8
result: 0 + 1 + 1 + 2 + 3 + 5 + 8 + 13 = 21
input X for fibonacci(x): 11
result: 0 + 1 + 1 + 2 + 3 + 5 + 8 + 13 + 21 + 34 + 55 = 89
input X for fibonacci(x):
-1
실행이 종료된 상태
```

Exited due to syscall 10
Simulator requested a breakpoint.

결론

MIPS 환경에서 과제를 구현해봄으로써 내부 메모리와 레지스터가 어떻게 동작하는지 이해하고, 컴파일러가 어떻게 프로그래밍 언어를 어셈블리어로 바꾸는지 이해하고 활용할 수 있게 되었습니다. MIPS 환경에서의 필요한 여러 명령어들과 MIPS 프로그래밍에 필요한 문법을 학습하는 기회가 되었습니다.