실습 1번

.data

std\_id: .asciiz "ID: "

std\_ret: .asciiz "Your ID is: "

.text

.globl main

main:

# system code 4: 문자열 찍어라.

# $a0에 'std\_id'을 가르킬 것이다.

li $v0, 4

la $a0, std\_id

syscall

# system code 5: 정수형으로 읽어라.

# $v0은 값이 들어갈 것이다.

li $v0, 5

syscall

#아래에 세개의 문장들을 같은 행동을 나타낸다.

# $s1 == $v0 -- $s1 <- $v0 or $0

# $s2 == $v0 -- $s2 <- $v0

# $s3 == $v0 -- $s3 <- $v0 add $0

#

# 이 세 문장들은 $v0의 다른 레지스터들로 변경될 것이다. ($s1-$s3 각각)

or $s1, $v0, $0

move $s2, $v0

add $s3, $v0, $0

# system code 4: 문자열로 찍어라.

# $a0은 std\_ret을 가르킬 것이다.

li $v0, 4

la $a0, std\_ret

syscall

# system code 1: 정수형 값을 찍는다.

# $a0은 $s1의 같은 값을 가질 것이다.

li $v0, 1

move $a0, $s1

syscall

# system code 10: 프로그램 종료

li $v0, 10

syscall

실습 2번

.data

std\_ask: .asciiz "Name: "

std\_ret: .asciiz "Your name is "

std\_name: .space 20 # 버퍼를 비워둔다.

.text

.globl main

main:

# system code 4: 문자열 프린트

# $a0은 std\_id 를 가르킬 것이다.

li $v0, 4

la $a0, std\_ask

syscall

# $a0 std\_name을 가르킬 것이다.

# 우리는 오직 20바이트만 넣을 것이다(19 아스키 코드 문자)

la $a0, std\_name

li $a1, 20

# system code 8: 문자열 읽어라

#

# $a0 : 문자열을 저장할 주소

# $a1 : 문자열 버퍼 길이

li $v0, 8

syscall

# system code 4: 문자열 읽어라

# $a0은 std\_ret을 가르킬 것이다.

li $v0, 4

la $a0, std\_ret

syscall

# system code 4: 문자열 읽어라

# $a0은 std\_name을 가르킬 것이다.

li $v0, 4

la $a0, std\_name

syscall

# system code 10: 프로그램 종료

li $v0, 10

syscall

실습 3번

.text

.globl main

main:

# 첫번째 계산.. 0xABCDEF \* 0xABCD

# $t1 <- 0xABCDEF

# $t2 <- $0 or 0xABCD

la $t1, 0xABCDEF

ori $t2, $0, 0xABCD

# $t1과 $t2은 곱셈이 될 것이다.

# 그 결과는 hi와 lo 레지스터에 저장될 것이다.

# hi레지스터의 값은 $s1으로 옮겨질 것이다.

# lo레지스터의 값은 $s2으로 옮겨질 것이다.

mult $t1, $t2

mfhi $s1

mflo $s2

# 두번째 계산.. 0xFFFFFFFF \* 0xABCD

# $t1 <- 0xFFFFFFFF

# st2 <- $s0 or 0xABCD

la $t1, 0xFFFFFFFF

ori $t2, $0, 0xABCD

# $t1과 $t2 곱셈이 될 것이다.

# 그 결과는 hi와 lo 레지스터에 저장될 것이다.

# hi레지스터의 값은 $s3으로 옮겨질 것이다.

# lo레지스터의 값은 $s4으로 옮겨질 것이다.

mult $t1, $t2

mfhi $s3

mflo $s4

# 마지막 계산.. 0xCDEF \* 0xCDEF

# $t1 <- 0xCDEF

# $t2 <- $0 or 0xCDEF

la $t1, 0xCDEF

ori $t2, $0, 0xCDEF

# $t1과 $t2 곱셈이 될 것이다.

# 그 결과는 hi와 lo 레지스터에 저장될 것이다.

# hi레지스터의 값은 $s5으로 옮겨질 것이다.

# lo레지스터의 값은 $s6으로 옮겨질 것이다.

mult $t1, $t2

mfhi $s5

mflo $s6

# system code 10: 프로그램 종료

li $v0, 10

syscall

실습 4번

.text

.globl main

main:

# 첫번째 계산.. 811 / 10

# $t1 <- 811

# $t2 <- $0 or 10

la $t1, 811

ori $t2, $0, 10

# $t1과 $t2는 나누어 질 것이다.

# 그 결과는 hi와 lo 레지스터에 저장될 것이다.

# hi레지스터의 값은 $s1으로 이동할 것이다.

# lo레지스터의 값은 $s2으로 이동할 것이다.

div $t1, $t2

mfhi $s1

mflo $s2

#두번째 계산.. -6 / 2

# $t1 <- -6

# $t2 <- $0 or 2

la $t1, -6

ori $t2, $0, 2

# $t1과 $t2는 나누어 질 것이다.

# 그 결과는 hi와 lo 레지스터에 저장될 것이다.

# hi레지스터의 값은 $s3으로 이동할 것이다.

# lo레지스터의 값은 $s4으로 이동할 것이다.

div $t1, $t2

mfhi $s3

mflo $s4

#세번째 계산.. 100 / 100

# $t1 <- 100

# $t2 <- $0 or 100

la $t1, 100

ori $t2, $0, 100

# $t1과 $t2는 나누어 질 것이다.

# 그 결과는 hi와 lo 레지스터에 저장될 것이다.

# hi레지스터의 값은 $s5으로 이동할 것이다.

# lo레지스터의 값은 $s6으로 이동할 것이다.

div $t1, $t2

mfhi $s5

mflo $s6

# system code 10: 프로그램 종료

li $v0, 10

syscall

과제

.text

.globl main

main:

# $s1 <- 0xABCDEF

# $s2 <- $0 or 10

la $s1, 0xABCDEF

ori $s2, $0, 10

# $s1과 $s2은 곱셈이 될 것이다.

# 그 결과는 hi와 lo 레지스터에 저장될 것이다.

# hi레지스터의 값은 $s3으로 옮겨질 것이다.

# lo레지스터의 값은 $s4으로 옮겨질 것이다.

mult $s1, $s2

mfhi $s3

mflo $s4

# $s1과 $s2는 나누어 질 것이다.

# 그 결과는 hi와 lo 레지스터에 저장될 것이다.

# hi레지스터의 값은 $s5으로 이동할 것이다.

# lo레지스터의 값은 $s6으로 이동할 것이다.

div $s1, $s2

mfhi $s5

mflo $s6

# 프로그램 종료

li $v0, 10

syscall

