CS 241 - Week 2 Tutorial Solutions

Assembly Basics and Assembly Language Programming

Fall 2018

1 Assembly basics

1.1 Assembling assembly language

Assemble the following program by first writing out its binary representation, then converting it to a hexadecimal representation.

```
slt $6, $1, $5
beq $6, $0, 1
```

Solution:

```
0000 0000 0010 0101 0011 0000 0010 1010
0001 0000 1100 0000 0000 0000 0000 0001
```

Next we need to convert them into hex. Every group of 4 binary digits corresponds to one hex digit: you might find it easiest to use a conversion chart, to convert to from binary to decimal and then decimal to hex, or to memorize the bit patterns. Regardless of which method you choose you should end up with:

```
.word 0x25302A
.word 0x10C00001
```

2 Assembly Language Programming

Problem 1 - A simple loop in MIPS

Recall that the factorial, $n!$, of n is given as follows:

$$\begin{aligned} &0! = 1 \\ &n! = n \cdot (n-1)! && n > 0 \end{aligned}$$

Write a MIPS program which takes a non-negative integer n in `$1` and stores $n!$ in `$3`. Solution:

```
        ;Initialize the answer ($3) = 1 and $11 = 1
        lis $3
        .word 1
        add $11, $3, $0

        ;Loop until $1 = 0
loop:   beq $1, $0, end

        ;$3 = $3 * $1
        mult $3, $1
        mflo $3

        ;Go to next index ($1 = $1 - 1)
        sub $1, $1, $11
        beq $0, $0, loop

end:    jr $31
```

Problem 2 - More loops in MIPS

Recall that the Fibonacci sequence can be defined as follows:

$$\begin{aligned} f_0 &= 0 \\ f_1 &= 1 \\ f_{n+2} &= f_{n+1} + f_n & n \geq 0 \end{aligned}$$

Write a MIPS program which takes a non-negative integer n in `$1` and stores f_n in `$3`.
Solution:

```
        ;$3 = f_i, $4 = f_{i+1}
        ;$11 = 1
        add $3, $0, $0
        lis $4
        .word 1
        add $11, $4, $0

        ;Loop until $1 = 0
loop:   beq $1, $0, end

        ;$5 = f_{i+1}
        add $5, $4, $0
        ;$4 = f_{i+2} = f_i + f_{i+1}
        add $4, $3, $4
        ;$3 = f_{i+1}
        add $3, $5, $0

        ;Go to the next iteration ($1 = $1 - 1)
        sub $1, $1, $11
        beq $0, $0, loop

end:    jr $31
```

Problem 3 - Arrays in MIPS

Thus far we've only written programs which accept two integers as arguments, but with `mips.array` we can also write programs which manipulate arrays. Write a MIPS program which accepts the address of an array in `$1` and its length in `$2` and stores the product of the numbers in the array in `$3`.

Solution:

```
        ;$2 = 4 * $2 + $1
        add $2, $2, $2
        add $2, $2, $2
        add $2, $2, $1

        lis $4
        .word 4

        lis $3
        .word 1

        ;Loop until $1 = $2, incrementing $1 by 4 every time
loop:   beq $1, $2, end

        ;$5 = *$1 = A[i]
        lw $5, 0($1)

        ;$3 = $3 * $5
        mult $3, $5
        mflo $3

        ;Go to next index
        add $1, $1, $4
        beq $0, $0, loop

end:    jr $31
```

Problem 4 - Basic I/O in MIPS

Recall that you can read from stdin and write to stdout by loading from or storing to addresses `0xffff0004` and `0xffff000c` respectively. Note that EOF is represented by -1, and otherwise a single byte will be read or written at a time.

Write a MIPS program which reads in two characters from stdin (you may assume EOF is not encountered) and prints out the character `1` if the first is less than the second, or `0` otherwise. It should then print a newline.

Solution:

```
;$27 is stdin, $28 is stdout
lis $27
.word 0xffff0004
lis $28
.word 0xffff000c

;$20 is the '0' character
lis $20
.word 48 ;0x30 in hex

;Load characters from stdin
lw $3, 0($27)
lw $4, 0($27)

;$3 = 1 if $3 < $4, 0 otherwise
slt $3, $3, $4

;Note that '0' + 0 = '0' and '0' + 1 = '1'
add $20, $20, $3

;Print the character to stdout
sw $20, 0($28)

;Newline is 10 = 0xA, so load and print it
lis $20
.word 10
sw $20, 0($28)

jr $31
```