CSE220 Fall 2014 - Homework 1

Due Friday 9/19/2014 @ 6pm

The goal of this homework is to become familiar with basic MIPS instructions and syscalls. You should be able to interpret pieces of binary data as different numerical formats: *signed-magnitude*, *1's complement*, and *2's complement*.

> ⚠ You **MUST** download the SB version of MARS posted on the PIAZZA website. **DO NOT USE** the MARS available on the official webpage. The SB version has a reduced instruction set and additional system calls you will need to complete the homework assignments.

Part 1: Read in Value from User

- Create a new MIPS program file.
- At the top of your program in comments put your name and id.

```
# Homework #1
# name: MY_NAME
# sbuid: MY_SBU_ID
...
```

- Add instructions to your .text section of the program to print a string prompt to the screen. The prompt should ask the user to enter an integer number.
    - You will need to create the string in the .data section.

```
Enter an integer number:
```

- Use the MARS syscall to read in the user entered integer value (syscall 5).

> For the remainder of the assignment we will call this entered value `x`. Remember, all numbers are stored in binary 2's complement format.
>
> No validation of user input is required in the assignment.

Part 2: Different Numerical Formats

In this part you will use basic MIPS instructions to read/manipulate the bits of `x` in order to re-represent the number in the binary representation of the other number formats.

For each format you will print a table which contains a label string, the value of the binary representation in that numerical format, the hexadecimal representation of the number, the binary value of the number, and the two's complement value of the number.

```
REP_LABEL:     REP_VALUE  0xHEX_VALUE BINARY_VALUE TWOS_COMPLEMENT_VALUE
```

Sample output examples:

```
Enter an integer number: 32896

2's complement:      32896  0x00008080 00000000000000001000000010000000  32896
1's complement:      32896  0x00008080 00000000000000001000000010000000  32896
Sign Magnitude:       32896  0x00008080 00000000000000001000000010000000  32896
Neg 2's complement: -32896  0xFFFF7F80 11111111111111110111111110000000 -32896
16-bit 2's comp:    -32640  0x00008080 00000000000000001000000010000000  32896

-- program is finished running --
```

```
Enter an integer number: -7

2's complement:     -7    0xFFFFFFF9 11111111111111111111111111111001 -7
1's complement:     -7    0xFFFFFFF8 11111111111111111111111111111000 -8
Sign Magnitude:      -7    0x80000007 10000000000000000000000000000111 -2147483641
Neg 2's complement:  7    0x00000007 00000000000000000000000000000111 7
16-bit 2's comp:    -7    0x0000FFF9 00000000000000001111111111111001 65529

-- program is finished running --
```

```
Enter an integer number:  697335811

2's complement:      697335811 0x29908003 00101001100100001000000000000011  697335811
1's complement:      697335811 0x29908003 00101001100100001000000000000011  697335811
Sign Magnitude:       697335811 0x29908003 00101001100100001000000000000011  697335811
Neg 2's complement: -697335811 0xD66F7FFD 11010110011011110111111111111101 -697335811
16-bit 2's comp:        -32765 0x00008003 00000000000000001000000000000011  32771

-- program is finished running --
```

- Begin by defining each label string in your .data section.
- For each format, you will need to:
    1. Print the label string (syscall 4)
    2. Manipulate the `x` value into the binary representation of the numerical format using bitwise and mathematical MIPS instructions
    3. Print the integer value of the row's binary representation `REP_VALUE` (new syscalls - see below)
    4. Print the hexadecimal value of the binary representation `0xHEX_VALUE` (syscall 34)
    5. Print the binary value of the representation `BINARY_VALUE` (syscall 35)
    6. Print the 32-bit binary value as a 2's complement number `TWOS_COMPLEMENT_VALUE` (syscall 1)

> ℹ When handling the 16-bit representation, set the upper 16 bits to zero.
>
> ℹ In the case of the 16-bit 2's complement number for step 6 you will print the entire 32-bit value using syscall 1. This is expected.

To print 1's complement, sign magnitude, and 16-bit values using MARS, we created additional syscalls. These syscalls are not standard and will not appear in the documentation.

description	**syscall**	**arguments**	**result**
print 1's comp	100	$a0 = value to print	
print SM	101	$a0 = value to print	
print 16-bit 2's comp	102	$a0 = value to print	

> ℹ syscall 102 will print out the value of the lower 16 bits of the register. For example if we have the value `0x00098001` => `00000000000010011000000000000001` syscall 102 will print out the value of `1000000000000001` in two's complement. The upper 16 bits will be disregarded.

How do I do #2: manipulate the 'x' value to get the different representations?

As discussed in lecture, bitwise operators can be used to mask, modify, and manipulate the bits in a register.

The following is a list of the MIPS instructions you should consider using to manipulate the bits of the register (you are not limited only to these): ADD, ADDI, AND, OR, NOR, NOT, NEG, SRL, SRA, SLL, SLT.

Ending your program

After translating to all the different representations you should have your program quit.

- Terminate your program with the exit syscall (10).

Part 3: What happens if…

Instead of reading an integer, suppose you were instead given 4 ascii characters. What table values would print out for the 5 formats?

- Place in the comments at the top of your .asm file the output table which would be printed by your program when “AbCd” is given.

> Remember each ASCII character fits in 1 byte, 4 characters fit in a word. Don’t forget about MARS endianess!

Hand-in instructions

See Sparky Submission Instructions on piazza for hand-in instructions.

> When writing your program try to comment as much as possible. Try to stay consistent with your formatting. It is much easier for your TA and the professor to help you if we can figure out what your code does quickly. If your program doesn’t work correctly the grader may be able to give you partial points, assuming he can read the code that you have written.