EE 3613: Computer Organization

Homework #2

Due Dates: (via blackboard) Hw #2A - Friday, September 18, 2020 by 11:59 PM
Due Dates: (via blackboard) Hw #2B - Friday, September 25, 2020 by 11:59 PM

Instructions:

1. All parts of the assignment (A and B) should be individually done, collaboration is not allowed.
2. For 2A, QtSPIM should not be used.
3. For 2B, please use QtSPIM.
4. For 2B, use the following naming convention, <last-name>_p1.s, <last-name>_p2.s and <last_name>_p3.s.
5. Attach a separate read_me.txt; when commenting the code, please re-run the code after commenting to check the syntax.
6. Please zip all the files before submitting; a single zipped file should be submitted.

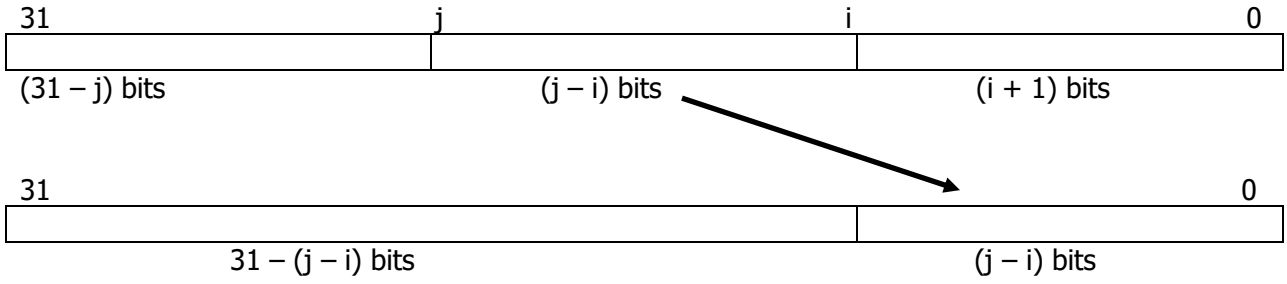
Hw 2A: HARDCOPY SUBMISSION (100 Points)

Problem 1 (10%):

(a) What binary number does the hexadecimal number represent 7fff $fffa_{hex}$? What decimal number does it represent?

(b) What hexadecimal number does this binary number represent: 1100 1010 1111 1110 1111 1010 1100 1110_{two}?

Problem 2 (10%):
Some computers have explicit instructions to extract an arbitrary field from a 32-bit register and place it in the least significant bits of a register. The figure below shows the desired operation:



Find the shortest sequence of MIPS instructions that extract a field for the constant values i = 5 and j =22 from register $t3 and places it in register $t0. (Hint: Can be done in 2 instructions)

Problem 3 (15%):
The following program tries to copy words from address in register $a0 to the address in register $a1, counting the number of words copied in register $v0. The program stops copying when it finds a word equal to 0. You do not have to preserve the contents of registers $v1, $a0 and $a1. This terminating word should be copied but not counted.

```
      addi $v0, $zero, 0     # Initialize count
Loop: lw $v1, 0($a0)         # Read next word from source
      sw $v1, 0($a1)         # Write to destination
      addi $a0, $a0, 4       # Advance pointer to next source
      addi $a1, $a1, 4       # Advance pointer to next destination
      beq $v1, $zero, Loop   # Loop if word copied != zero
```

(a) There are bugs in this program, turn in a bug-free version.

(b) How many bytes are required to execute this program? How many instructions are ALU instructions?

Problem 4 (15%):
What is the final state of the memory after executing these instructions?

```
lhu $3, 100($0)
lb $4, 102($0)
sw $3, 100($0)
sh $4, 102($0)
```

Memory Location	Value Stored
100	0xFF
101	0xFF
102	0x77
103	0xFF

Problem 5 (20%):
In this exercise, you will explore 32-bit constants in MIPS. For the following problems, you will be using the binary data in the table below:

a) 1010 1101 0001 0000 0000 0000 0000 0010
b) 1111 1111 1111 1111 1111 1111 1111 1111

1) Write the MIPS code that creates the 32-bit constants listed above and stores that value to register $t1
2) If the current value of the PC is 0x00000000, can you use a single jump instruction to get to the PC address as shown in the table above?
3) If the current value of the PC is 0x00000600, can you use a single jump instruction to get to the PC address as shown in the table above?
4) If the current value of the PC is 0x00400600, can you use a single jump instruction to get to the PC address as shown in the table above?
5) If the immediate field of a MIPS instruction was only 8 bits wide, write the MIPS code that creates a 32-bit constants listed above and stores that value to register $t1. Do not use the `lui` instruction.

Problem 6 (20%):

In MIPS assembly, write an assembly language version of the following C code segment:

```
for (i = 0; i < 98; i ++) {
      C[i] = A[i + 1] - A[i] * B[i + 2]
}
```

Arrays A, B and C start at memory location 0xA000, 0xB000 and 0xC000 respectively. Try to reduce the total number of instructions and avoid multiply instructions.

Problem 7 (10%):

This C version of a *case* statement is called a *switch* statement. The following C code chooses among four alternatives, depending on whether k has the value 0, 1, 2, or 3.

```
switch (k) {
      case 0:f = i + j; break; /* k = 0 */
      case 1:f = g + h; break; /* k = 1 */
      case 2:f = g – h; break; /* k = 2 */
      case 3:f = i – j; break; /* k = 3 */
}
```

Assume the six variables f through k correspond to six registers $s0 through $s5 and that register $t2 contains 4. What is the corresponding MIPS code? [Hint: Use the *switch* variable k to index a jump address table, and then jump via the value loaded. We first test K to be sure it matches one of the cases ($0 \leq k \leq 3$); if not, the code exits the *switch* statement.]

Hw 2B: SOFTCOPY SUBMISSION (100 Points)

Programming 1 (35%): Write a SPIM program that will compute the transpose of the matrix stored in row major order starting at the location labeled `Original` in the starter template below. The transposed matrix should be stored starting at the location labeled `Second`. The arrays we use will be 4x4 elements. Here is the program template to help you get started.

```
.data
strA:       .asciiz "Original Array:\n "
strB:       .asciiz "Second Array:\n: "
newline:    .asciiz "\n"
space :     .asciiz " "

# This is the start of the original array.
Original:   .word 200, 270, 250, 100
            .word 205, 230, 105, 235
            .word 190, 95, 90, 205
            .word 80, 205, 110, 215
# The next statement allocates room for the other array.
# The array takes up 4*16=64 bytes.
#
Second: .space 64

.align 2
.globl main
.text
main: # Your fully commented program starts here.
```

The program should include a block of code that prints the transposed matrix to the screen in row order. Your program should be well commented.

Programming 2 (30%): Write a MIPS code to compute the sum of N numbers such that each sum skips over certain numbers. Ask the user for how many numbers to skip (give option of 1-4) and then proceed to compute the sum.

Sum(x1) = N1 + N2 + N3 + …
Sum(x2) = N1 + N3 + N5 + ….
Sum(x3) = N1 + N4 + N7 + ….
Sum(x4) = N1 + N5 + N9 + ….

Sum(x1) determines the sum of all numbers; Sum(x2) determines the sum of numbers skipped by 2, Sum(x3) determines the sum of numbers skipped by 3 and so on. Assume the numbers as given below where N1 = 100, N2 = -7, N3 = 11, N4 = 25 and so on.

```
.data
.strA:      .asciiz "Please enter your choice to skip numbers (1-4)\n"
Numbers:    .byte 100, -7, 11, 25, -66, 99, -1, 34, 12, 22, -2, -7, 100,
11, 4, 67, 2, -90, 22, 2, 56, 3, -89, 12, -10, 21, 10, -25, -6, 9, 111,
34, 12, 22, -2, -17, 100, 111, -4, 7, 14, -19, -2, 29, 36, 31, -79, 2

.globl main
.text
main: # Your fully commented program starts here.
```

Programming 3 (35%): Some recursive procedures can be implemented iteratively without using recursion. Iteration can significantly improve performance by removing the overhead associated with procedure calls.

Write a MIPS procedure to compute the *n* th Fibonacci number F(n) where

```
F(n) = 0, if n = 0; 1, if n = 1;
F(n-1) + F(n-2), otherwise.
```

Base your algorithm on the *recursive* process:

```
int fib(int n){
if (n == 0)
      return 0;
else if (n == 1)
      return 1;
else
      return fib(n-1) + fib(n-2);
```

Requirements: Ask the user for "n" Fibonacci number; then display all the numbers upto "n". For example:

- Please enter the nth Fibonacci number to be displayed:
- 6
- 0, 1, 1, 2, 3, 5

For example:

- Please enter the nth Fibonacci number to be displayed:
- 8
- 0, 1, 1, 2, 3, 5, 8, 13