CS 250 Spring 2017 Homework 06
Due 11:58pm Wednesday, March 01, 2017
Submit your typewritten file in PDF format to Blackboard.

1. What minimum modification to Figure 6.4 to re-design the computer of Chapter 6 as an architecture using 64-bit instructions? Given that instruction size and address size often match, what additional change should be made to Figure 6.4?
   Change the Program Counter and Adder to be 64 bit

2. Design an instruction for the processor of Figure 6.9 called *Branch Indirect*, with the mnemonic BRI. BRI sets the Next_Instruction_Pointer equal to the contents of data memory at the sum of an immediate value and a register.
   a. In the style of Figure 6.2, show all information necessary to define BRI.

Operation	Reg A	Reg B	Dst Reg	Offset
00101	4-bit value	4-bit address pointer	4 bit-unused	15-bit unused

   b. Execution of a BRI instruction will create what values for the register unit control signals?
      RegDst = 1

   c. What operation will be performed by the ALU for BRI?
      Add

   d. What control signals will be created by BRI for data memory?
      Writeback

   e. Which inputs (upper or lower) will be selected by BRI for multiplexers M1, M2, and M3?

M1	M2	M3
Upper	Upper	Upper

   f. What other Chapter 6 instruction has this same multiplexer selection pattern?
      Branch

3. How many control signals are generated when an instruction is decoded in Figure 6.9? Assume that the ISA may contain 32 instructions.
   13 control signals

4. Assume that a single chip implements M2 in Figure 6.9. How many pins does this chip have?
   15bits + 32bits = 47 bits. **47 pins**

5. Define three new instructions in the format of Figure 6.2 as follows. JSR, jump to subroutine, has an opcode field of 00101 and executes by saving the current

default_next_instruction_address in a stack data structure in memory and branching to default_next_instruction_address + Offset. RET, return from subroutine, has opcode 00110 and has automatic access to the address of the top of stack in memory and causes the contents of the top of stack in memory to become the value of next_instruction_address in the fetch circuitry. AND, bitwise logical AND, has opcode 00111.

a. Write the machine code for each line of assembly in the following program snippet. A "snippet" is a few lines of code that may not make logical sense, likely due to missing context. The symbol … confirms that the assembly program exists within a larger context, but that context is not to be part of your answer. Label each line of machine code with its hexadecimal address. **Use a constant-wide font for your answer and put a space between each field in the format of Figure 6.2 to aid readability, knowing that there are no spaces in machine code.** If a format field can contain any bit string, then fill that field with 1. Instead of using … in your answer to denote the unknown contents of memory between main: and MiddleBytes: state on that line of your answer how many intervening instructions could fit in the memory region where the content has not been specified in the assembly program. Note that all registers are global, and thus visible at all times to all instructions.

```
0x003FFFFC                …
0x00400000   main:        JSR MiddleBytes     ; call subroutine MiddleBytes
0x00400004                STORE r1, 0(r5)     ; Memory[r5+0] <- r1
0x00400008                ADD r2, r3, r4      ; r2 <- r3 + r4
0x0040000C                JSR MiddleBytes     ; call subroutine MiddeBytes
  …                        …
0x00400024   MiddleBytes: AND r1,r2,4080      ; r1 <- r2 AND 0x0FF0
0x00400028                RET                 ; return from MiddleBytes
0x0040002C                …
```

```
0x003FFFFC: 1 intervening instruction
0x00400000: 00101000000000000111111111111111
0x00400004: 00011111111110000000000000000000
0x00400008: 00001111111111111000000000000000
0x0040000C: 00101000000001111111111111111111
…         : 1 intervening instruction
0x00400024: 00111111111111111000000000000000
0x00400028: 00110000000000000000000000000000
0x0040002c: 1 intervening instruction
```

b. What are the value(s), if any, of all symbolic addresses in the assembly language of part a. of this question?
MiddleBytes: 0x00400000
Main: 0x00400024

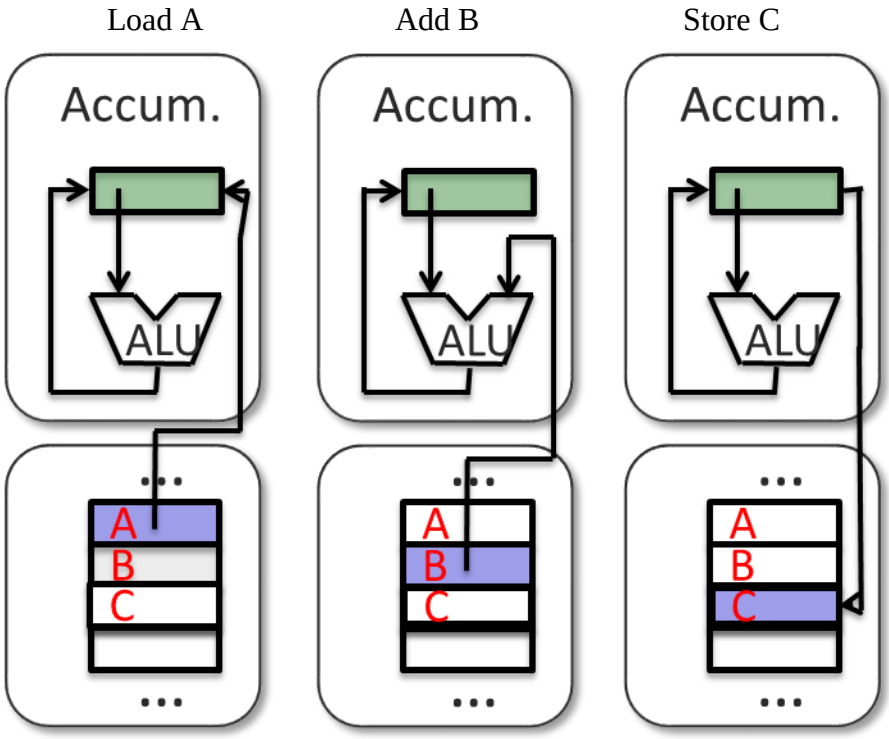
c. What are the immediate value(s), if any, in the assembly language of part a. of this question? State any value as a tuple of the form (mnemonic, 0x…). If the value does not fit exactly into an integer number of hexadecimal digits, assume that any extra

bits in the hexadecimal notation are zeros. If there are no immediate values, clearly so state in your answer.
0x0FF0

d. Are any of the instructions in your machine code position dependent? If not, why not, and is so, which instructions(s) and why?
RET, return has to know why instruction it needs to return to and must be set according to its position from the caller.

e. Did generation of the machine code in your answer to part a. rely on the assembler being two-pass? Describe the action of the two passes with respect to generating the immediate values, if any listed in your answer to part c. of this question.

6. Using Lecture 13, Slide 15 as a reference, draw a sequence of three diagrams showing the movement of data through a 1-address machine executing the three lines of assembly corresponding to C=A+B. You answer should contain one diagram for each line of code. The architectural part of each of the three diagrams should be identical. Then, each diagram also shows the location of each data item mentioned in the corresponding line of code and has arrows showing all data item movement(s) resulting from the execution of that line of code. See Lecture 13 Slides 6 through 9 for an example of the identical question answered for a 0-address architecture.
CODE: Load A, Add B, Store C



7. Which addressing modes of Figure 7.6 are
   a. impossible, and which are possible, for a machine with 32-bit instructions and a 32-bit address? Give a reason for any impossible mode.

Direct and Indirect memory reference is impossible because you can't fit the 32bit memory address within the 32bit instruction.
Immediate value, Direct register reference, and Indirect through a register are possible.

b. perhaps possible for a machine with 64-bit instructions and 16 Gbytes of byte-addressed memory? Give reasons.

You need a 27bit address to map 16 Gbytes. A 27 bit address will fit in the parameters of a 64bit instruction.