BIRZEIT UNIVERSITY

Computer Organization & Assembly Language

ENCS 336

Homework #1

Submission deadline: Saturday Oct 10, 2015

Instruction format:

Q(1)

Suppose a datapath has three operand busses (two source, one destination), 45 instruction types, and 32 registers where each register is 16 bits wide. Immediate operands can be in the range of ±128K.

Design an instruction format for instructions that have one operation, one destination register and two source registers. Label the fields and minimum number of bits need for each field.

Opcode: ? bits	**Destination Register: ? bits**	**Source 1 Register: ? bits**	**Source 2 Register: ? bits**

Q(2)

Suppose we are designing an instruction set architecture with 32-bit instructions and 26 different opcodes. The register file contains 128 registers. One of the instruction types we would like to support specifies an opcode, a destination register, and two immediate source values. What is the minimum number of bits that are needed to specify each field?

Opcode: ? bits	**Destination Register: ? bits**	**Immediate Value: ? bits**	**Immediate Value: ? bits**

Q(3)

Suppose we are designing an instruction set architecture with 28-bit instructions and 44 different opcodes. Immediate operands can be in the range of ±512. How many registers can this datapath have?

Opcode	**Destination Register**	**Source Register**	**Immediate**

Endianess

Q4) For the following data structures, draw the big-endian and little-endian layouts

1. **struct** {
   double i; //0x1112131415161718
   };

2. **struct** {
   int i; //0x11121314
   int j; //0x15161718
   };

3. **struct** {
   short i; //0x1112
   short j; //0x1314
   short k; //0x1516
   short l; //0x1718
   };

Stack-Oriented Instructions

Q5)

a) Convert the expression A + B - C to postfix notation. Show the steps involved.
b) Is the result equivalent to (A + B) - C or A + (B – C) ?
c) Does it matter?

Q6) Convert the following formulas from infix to reverse Polish:

a) A + B + C + D + E

b) (A + B) * (C + D) + E

c) (A * B) + (C * D) + E

d) (A - B) * (((C - D * E)/F)/G) * H

Shift Left/Right Operations

Q7)

a) Compute Arithmetic and Logical Left/Right Shift on the following numbers using 9-bit binary representation (two's Complement).

   a. Number: $(27)_{10}$, Shift Left 3 positions.

   b. Number: $(155)_{10}$, Shift Left 3 positions.

   c. Number: $(-22)_{10}$, Shift Left 2 positions.

   d. Number: $(123)_{10}$, Shift Right 3 positions.

   e. Number: $(-167)_{10}$, Shift Right 4 positions.

b) It was stated that Arithmetic Shift Left correspond to multiplication by a power-of-2 integer if there is no overflow. Demonstrate this fact for the above numbers (a, b, c). Does this fact hold for Logical Shift Left? For cases where overflow occurs, how many additional bits we need to avoid overflow and make the shift operation result correspond to multiplication operation?

c) It was stated that Arithmetic Shift Right correspond to division by a power-of-2 integer. Demonstrate this fact for the above numbers (d, e). Does this fact hold for Logical Shift Right? In what way are numbers rounded using arithmetic right shift (e.g., round toward $-\infty$ or toward $+\infty$)?

Assembly Programs

Q8) Compare Zero-, one-, two-, and three-address machines by writing programs to compute the following expression:

```
(A - B) * (((C - D * E)/F)/G) * H
```

0 Address	1 Address	2 Address	3 Address
PUSH M	LOAD M	MOVE (X ← Y)	MOVE (X ← Y)
POP M	STORE M	ADD (X ← X + Y)	ADD (X ← Y + Z)
ADD	ADD M	SUB (X ← X − Y)	SUB (X ← Y − Z)
SUB	SUB M	MUL (X ← X × Y)	MUL (X ← Y × Z)
MUL	MUL M	DIV (X ← X/Y)	DIV (X ← Y/Z)
DIV	DIV M		

Q9) Given the Assembly code sample below:

a) Add comments to each instruction showing what does it do?

b) How many times the code between **`Loop`** Label and **`finish`** Label will be executed (i.e. loop iterations)?

c) Assuming **`'a0'`** is input register (16-bit) holding value 0xFA27 (in Hexadecimal), and **`'v0'`** is output register, compute the output value in **`'v0'`**.

d) If the processor doesn't support **`move`** instruction, how we could re-write the code below to remove **`move`** instructions? With what other instructions we can replace them?

```
        move r0, 0
        move r3, 0
        move r1, 1
loop:   BRE  r1, 1024, finish    ; BRE: Branch if equal
        add  r0, r0, r1
        and  r2, r0, a0
        add  r3, r3, r2
        ashl r1, r1, 1  ; ashl: arithmetic shift left
        BR loop         ; BR: Branch
finish:
        move v0, r3
```

Addressing Modes

Q10) An address field in an instruction contains decimal value 14. Where is the Corresponding operand located for

a) Immediate addressing?
b) Direct addressing?
c) Indirect addressing?
d) Register addressing?
e) Register indirect addressing?

Q11) A PC-relative mode branch instruction is 3 bytes long. The address of the instruction, in decimal, is 256028. Determine the branch target address if the signed displacement in the instruction is –31.