CSE 321a: Computer Organization (I)

Third Year, Computer & Systems Engineering

Assignment #3

Due date: **Tuesday, December 27th, 2016**

1. Consider an 8-bit stack machine with the following characteristics:
   - Memory: byte-addressable, 8-bit addresses.
   - Data words (8-bit): logical, unsigned integers, or signed integers (sign-and-magnitude).
   - Registers (8-bit): PC, SP, and four general purpose registers (R0, R1, R2, and R3).
   - Instructions (variable-length): zero-operand or one-operand.
     - Zero-operand instructions (8-bit): 8-bit opcode.
     - One-operand instructions (16-bit): 6-bit opcode followed by one 10-bit operand.
       - Operands (10-bit): 2-bit addressing mode (AM) followed by 8-bit value/address (VA).
       - Addressing-mode (2-bit): 00 ↔ immediate, 01 ↔ register, 10 ↔ register-indirect, 11 ↔ base-register (where register number is encoded by the least significant two bits of VA).
     - Some zero-operand mnemonics and their corresponding opcodes (**decimal**):
       - NEGATE ↔ 67, ADD ↔ 75, MULTIPLY ↔ 115.
     - Some one-operand mnemonics and their corresponding opcodes (**decimal**):
       - PUSH ↔ 23, POP ↔ 24.
       - SKIP ↔ 29 (skip *n* bytes, where *n* is the unsigned value of the operand)
       - SKIPZ ↔ 30 (pop the stack top and skip *n* bytes if the popped value equals zero)

(a) Suppose in this instruction set, the most significant bit of the opcode is used to distinguish between zero-operand and one-operand instructions. Calculate the maximum number of opcodes that can be included in this instruction set? Justify your answer and suggest another way for encoding the opcodes in order to maximize their number without changing the instruction format.

(b) Translate the following C-code snippet into the assembly language of this machine:

```
/* x is array of single-byte integers */
if (x[2] == 0)
  x[3] = -x[3];
else
  x[3] = 50 + x[2]*(-x[3]);
```

Assume that register R1 contains the address of the first element in the array (*i.e.*, `x[0]`). Your assembly program must start with the following instruction: "`PUSH R1(2)`".

(c) Assume your assembly program (from part (a)) was compiled to the machine language and stored in the main memory starting at location 1D. Fill up the following table with the rest of the machine program:

Address (Hexadecimal)	**Contents (Hexadecimal)**
1D	5F
1E	09
1F	...
…	...

(d) Show, using the table below, the execution trace of the machine program (from part (b)) by filling in the contents of every register and memory location **after the execution** of every instruction. All values are in hexadecimal.

Instruction	**PC**	**SP**	**R1**	**Memory Locations**			
				72	**73**	**CE**	**CF**
Initially	1D	D0	70	06	85	BE	EF
5F09	1F	CF	70	06	85	BE	06
...	...	...	...	...	...	...	...

2. Describe the sequence of micro-operations (and show the steps) required for a CPU (in which each register has a dedicated connection with every other register and every ALU input) to execute each of the following instructions:

(a) `POP (2000)`

(b) `BSA @(-100)                /* Branch and save address */`

(c) `MOVE (700)(R2),500(R1)-   /* 1st operand is destination */`