CISC 260 Machine Organization and Assembly Language

Practice Midterm Exam

This is an open-note exam. You are allowed to use notes. You are NOT allowed to use electronic devices except standard calculators.

1. [25 points] Data representations and arithmetics

a. Convert 33_{ten} into a 8-bit two's complement binary number.

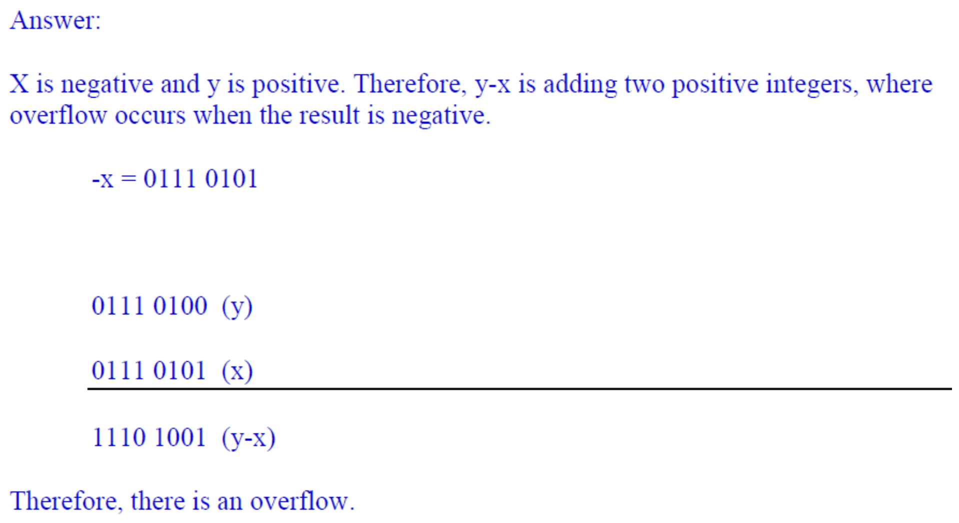
Answer: 0010 0001

b. What decimal number does the following two's complement 8-bit binary number represent?

1100 1010 $= -54_{ten}$

c. Is there an overflow for an 8-bit machine when subtracting a two's complement integer x from a two's complement integer y as given below? Show your work.

x = 1000 1011 and y = 0111 0100



Answer:

X is negative and y is positive. Therefore, y-x is adding two positive integers, where overflow occurs when the result is negative.

-x = 0111 0101

0111 0100 (y)

0111 0101 (x)

1110 1001 (y-x)

Therefore, there is an overflow.

d. Show the negation of the following integer in two's complement.

$X = 1101\ 0110\ 0111\ 0101_{two}$

Answer: $-x = 0010\ 1001\ 1000\ 1011_{two}$

e. In multiplying the following two integers A and B , how many times the (properly shifted) multiplicand is added to the (intermediate) product C = A x B if the multiplication is implemented using the shift-add algorithm?

A = 1010 0101

B = 0110 1001

If you are using the booth algorithm, the result would be different—only execute when the 2 bits from multiplier are 01, 01,00,10,01,10,11,01

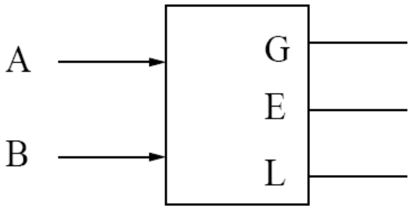
Answer: 4.

so it will be executed 3 times

The number of entering will be different from the number of executing (greater than the latter by 1—entering and done)

2. [20 points] Boolean Logic and Gates

A comparator circuit has two 1 bit inputs A and B and three 1 bit outputs G (greater), E (Equal) and L (less than)



G = 1, if A > B 0, otherwise	E = 1, if A = B 0, otherwise	L = 1, if A < B 0, otherwise

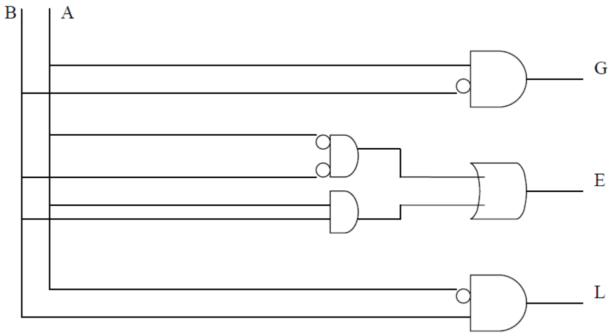
a. Fill out the truth table

A	B	G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

b. Write the Boolean expression in canonical form corresponding to the above truth table

G = A &~B; E = (~A&~B) | (A & B); L = ~A & B

c. Implement the circuit by using AND, OR and NOT gates. Draw the wiring diagram.



3. [25 points] ARM Instruction set

a. If register r4 has a value 0x f000 000c , what is the value in r0 as the result of running the following ARM assembly language program ?

never convert from decimal to hex, always have binary in the middle

```
     CMP r4, #0  result= r4-0
     BLE L1                    NZCV     result<=0? (signed integer)
     MOV r5, #1
     B L2
L1:  MOV r5, #2
L2:  MOV r0, r5
```

Write the value in decimal: **r0 = 2**

b. For the following ARM assembly code,
Address code

-----------------------------**BL: the instruction that can reset the values in r14**

BL does 2 things: 1. put the address of next instruction into r14; after the subroutine, 2. the program will return to the caller—where the procedure is called

pc=r15, the value of r15 changes every cycle

```
0x0000 1000        Main: MOV r4, #5
0x0000 1004              BL FOO
0x0000 1008              SWI 0x11
0x0000 100C         FOO: MOV r5, #1
0x0000 1010          L1: CMP r4, #0
0x0000 1014              BLE L2
0x0000 1018              MUL r6, r5, r4
0x0000 101C              MOV r5, r6
0x0000 1020              SUB r4, r4, #1
0x0000 1024              B L1
0x0000 1028         L2:  MOV r0, r5
0x0000 102C              MOV pc, r14
```

i. When the program halts, what are the values in the following registers?
r0 = 120
r14 = 0x0000 1008
r15 = 0x0000 1008

ii. How many time has the instruction “MUL r6, r5, r4” been executed?

5

iii. What does the program compute?

The program computes factorial for the integer stored in r4, in this case, it is 5! = 120.

4. [30 points] ARM Assembly programming

The following is a C function that takes an integer n > 0 and returns 1 + ... + n.

```
int sum_to (int n) {
     if (n<=1) return 1;
     else
          return n + sum_to(n-1);
}
```

a) You are asked to translate the program into ARM assembly code. You may assume that n is in r0, and write the returned value in r1.

b) If n = 5, how many activation frames are pushed onto the stack during the execution of the above program.

Answer:

a)

```
sum_to: sub sp, sp, #8
        str lr, [sp,#0]
        str r0, [sp,#4]
        cmp r0,#1
        bgt else
        mov r1, #1
        add sp, sp, #8
        mov pc, lr
else: sub r0, r0, #1
        BL sum_to
        mov r2, r1
        ldr r1, [sp, #4]
        ldr lr, [sp, #0]
        add sp, sp, #8
        add r1, r2, r1
        mov pc, lr
```

b) 5