Computer Organization (CDA – 3103)
Spring 2005
Lab # 5: Solution of Midterm Exam

Question 1: Convert following number into IEEE single precision floating point number (biased).
$(315.515625)_{10}$

Answer : Convert integer and fractional part of this number into binary number.

2	315
2	157 – 1
2	78 – 1
2	39 – 0
2	19 – 1
2	9 – 1
2	4 – 1
2	2 – 0
2	1- 0
2	0- 1

2 * 0.515625	0.03125 + 1
2 * 0.03125	0.0625 + 0
2 * 0.0625	0.125 + 0
2 * 0. 125	0.25 + 0
2 * 0.25	0.5 + 0
2 * 0.5	0 + 1

$(315.515625)_{10}$ = $(100111011.100001)_2$ = $(1.00111011100001 * 2^8)_2$

Sign-bit = 0
Exponent = $(8)_{10} + (127)_{10} = (135)_{10} = (10000111)_2$
Mantissa = 00111011100001000000000

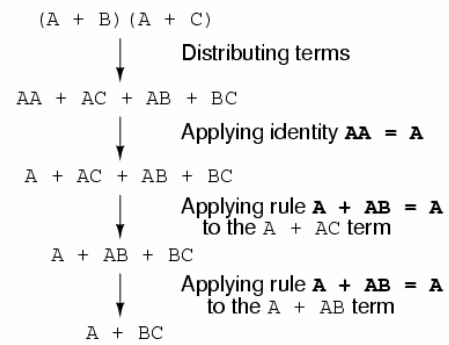
Sign	Exponent								Mantissa																							
0	1	0	0	0	0	1	1	1	0	0	1	1	1	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Question 2: Simplify following Boolean expressions using Boolean algebraic identities.

a) **A + AC + B (A + C) + C**

= A + AC + B A + BC + C (distributive law)
= A + AC + B A + BC + 1.C (identity w.r.t add operation)
= A + AC + B A + C (B + 1) (distributive law)
= A + AC + B A + C (B + 1 = 1)
= A + B A + AC + C (rearranged terms)
= A (1 + B) + C (A+1) (same as abve)
= A + C

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



Question 3: Convert following C program into assembly.

```
int Fibonacci(int n)
{
        int last, secondlast;
        int count = 2;
        int sum = 2;
        int temp;

        secondlast = 1;
        last = 1;

        while( count < n )
        {
                temp = secondlast + last;
                secondlast = last;
                last = temp;
                sum += temp;
                count ++;
        }
}
```

Answer:

For every variable, we will use a register. Suppose given variable to register mapping is used:

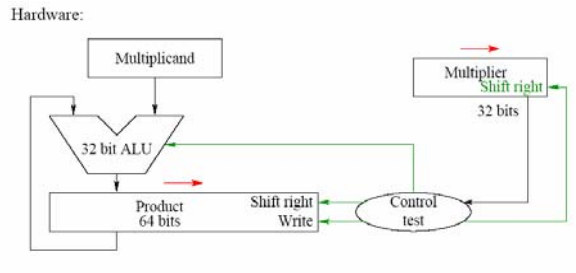
last = AX , secondlast = BX , count = CX , sum = DX , temp = BP , n = SI

```
        mov CX , 2
        mov DX, 2
        mov BX , 1
        mov AX , 1

LABEL_1

        cmp     CX,SI                   // while( count < n )
        jz      LABEL_2                 // while( count < n )           {conditional jump that check zero flag}

        mov BP,BX                       // temp = secondlast + last;
        add BP, AX                      // temp = secondlast + last;
        mov BX,AX                       // secondlast = last;
        mov AX,BP                       // last = temp;
        add DX,BP                       // sum += temp;
        add CX,1                        // count ++;

        jmp LABEL_ 1

LABEL_2
```

Question 4: Using floating point addition, show the steps (4 steps) for adding 0.5 and -0.4375.

Answer:See book page 198 – 201

Question 5: Describe how the multiplication circuit works by tracing an example.



Answer:
This circuit has been designed for 32-bit multiplication operation. We can understand its functionality by scaling it down to some smaller numbers; say 4-bit numbers

Suppose we have two numbers

Multiplicand = 1011
Multiplier = 1001

Initially
Product register (8-bit) = 00000000.
Multiplicand register (8-bit) = 00001011.
Multiplier register (8-bit) = 1001.

Step 1:
Control unit will test the least significant bit of multiplier which in this case is 1.
Add multiplicand 00001011 in product. Now product is 00001011.
Right shift multiplier register by 1. Now multiplier is 0100.
Left shift multiplicand register by 1. Now multiplicand is 00010110.

Step 2:
Control unit will test the least significant bit of multiplier which in this case is 0.
Right shift multiplier register by 1. Now multiplier is 0010.
Left shift multiplicand register by 1. Now multiplicand is 00101100.

Step 3:
Control unit will test the least significant bit of multiplier which in this case is 0.
Right shift multiplier register by 1. Now multiplier is 0001.
Left shift multiplicand register by 1. Now multiplicand is 01011000.

Step 4:
Control unit will test the least significant bit of multiplier which in this case is 1.
Add multiplicand 01011000 in product. Now product is 01100011.
Right shift multiplier register by 1. Now multiplier is 0000
Left shift multiplicand register by 1. Now multiplicand is 10110000

END

Final Status : Product register contains the product of these two number = 01100011
Multiplier register = 0000
Multiplicand (shifted in left half) = 10110000

Question 6: Convert code1.asm to machine code, write it using hexadecimal numbers.

rrr:

000	:	AL or AX
001	:	CL or CX
010	:	DL or DX
011	:	BL or BX
100	:	AH or SP
101	:	CH or BP
110	:	DH or SI
111	:	BH or DI

Name	**Regs**	**Description**	
PUSH	Reg Word	01010rrr	Push operand onto stack
	Seg	00sss110	Push operand onto stack
POP	Reg Word	01011rrr	Pop a word from the stack
	Seg	00sss111	Pop a word from the stack
ADD	Reg, Reg	0000001woorrrmmm	Add integers
	Acc, Imm	0000010w	Add integers
	Reg, Imm	1000000woo000rrr	Add integers

```
//code1.asm
        push ax
        push cx
        add a1,10d
        add c1,20d
        pop cx
        pop ax
END
```

Answer:

//code1.asm					
push ax	01010rrr	=	01010000	=	0x50
push cx	01010rrr	=	01010001	=	0x51
add a1,10d	00000100w 10d	=	00000100 00001010	=	0x040A
add c1,20d	1000000woo000rrr 20d	=	10000001 11000001 0001 0100	=	0x 81C114
pop cx	01011rrr	=	01011001	=	0x59
pop ax	01011rrr	=	01011000	=	0x58
END					