15CSCI10I
Computer Architecture
Assignment 1

Exercise 1

a) The following memory units are specified by the number of memory cells times the number of bits per memory cell. How many address lines and data lines are needed in each case?

i. 4K×32

ii. 16K×8

Solution:

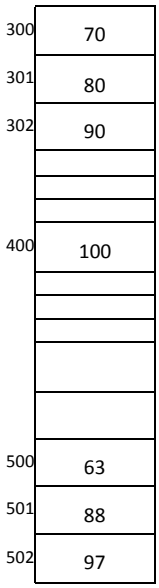
Question	Memory	Address lines	Data lines
i.	$2^2 \times 2^{10} = 2^{12}$ cells	12	32
ii.	$2^4 \times 2^{10} = 2^{14}$ cells	14	8

[2 marks]

b) Given the table representing the content of a memory unit:

i. In order to retrieve the contents of address 300, state what should be the values of special purpose registers AR and DR.

ii. If it is required to store number 70 in address 502, what should be the values of special purpose registers AR and DR.



Solution:

a) AR = 300 , DR = 70 (after retrieval)

b) AR = 502, DR = 70.

[2 marks]

Exercise 2:

Write ARM7 assembly language program that is equivalent to the following Java code:

```
public static void main(String[] args) {
    int product =1;
    int i = 0;
    while (i<10)
    {
       int j=0;
       while (j<5)
       {
          product+= i*j;
          j++;
       }
       i++;
    }

  }
```

[8 marks]

Solution:

```
        AREA Exercise2, CODE
        ENTRY
        MOV R0,#1

OUT CMP R1, #10
        BGE EXT
        MOV R2, #0
```

```
IN      CMP R2, #5
        BGE CNT

        MUL R3,R1, R2
        ADD R0, R3
        ADD R2,      #1
        B IN

CNT     ADD R1, #1
        B OUT

EXT
        END
```

Exercise 3:

Assume an initial hexadecimal value in Register R1= 0xFD, write an ARM7 assembly program to perform the following operations on the R1 consecutively using logical instructions:

i) Clear bits 0, 4 and 6.

ii) Complement bits 1, 2, and 7.

iii) Insert hexadecimal digit A instead of the bits from bits 4 to bits 7.

[6 marks]

Solution:

```
        AREA Exercise3, CODE
        ENTRY

        MOV R1,#0xFD
        AND R1,#0xAE
        EOR R1,#0x86
        AND R1,#0x0F
        ORR  R1,#0xA0

        END
```

Exercise 4:

Write an ARM7 Assembly program to perform an addition of the odd numbers from 1 to 10 and store it in R3, and multiply the even numbers except zero and store it in R5 (given that R5 is initially 1), your code should involve checking whether the number is even or odd using logical instructions. Note that the loop counter must start with value zero.

[10 marks]

Solution:

```
        AREA Exercise4, CODE
        ENTRY

        MOV R5,#1

LOOP CMP R0, #0
         BEQ INC
         CMP R0, #10
         BEQ EXT
         ANDS R2, R0, #0x01
         BEQ even
         ADD R3, R0
         B INC

even  MUL R6, R0, R5
        MOV R5, R6

INC  ADD R0, #1
       B LOOP

EXT
        END
```

Exercise 5:

A given program consists of 200-instruction loop that is executed 43 times. If it takes 32,000 cycles to execute the program on a given system, what are that system's CPI and IPC values for the program? What is your comment on using these 2 metrics to measure a system performance?

[2 marks]

Solution:

Total number of instruction executed = 200x43 = 8600

Total number of clock cycles for the program = 32000

CPI = Total number of clock cycles / instruction count = 32000/8600 = 3.72

IPC = Instruction count/ Total number of clock cycles = 8600/32000 = 0.268

Comment: When using IPC and CPI to compare systems, it is important to remember that high IPC values indicate that the reference program took fewer cycles to execute than low CPI values. Thus, a large IPC tends to indicate good performance; while a large CPI indicates poor performance.

Exercise 6:

Assume that a given architecture does not have a hardware support for division, so divisions have to be done through repeated addition and subtraction. If it takes 400 cycles to perform a division in software and 8 cycles to perform a division in hardware.

What's the overall speedup from hardware for division if a program spends 20% of its time doing divisions?

[4 marks]

Solution:

Speed up = 400/8 = 50 (ratio of time to do a division without the hardware to time with the hardware).

Using Amdahl's law
Frac used = 0.2
Frac unused = 1 - 0.2 = 0.8
Speed up = 1 / [0.8 + (0.2/50)] = 1.24

Exercise 7:

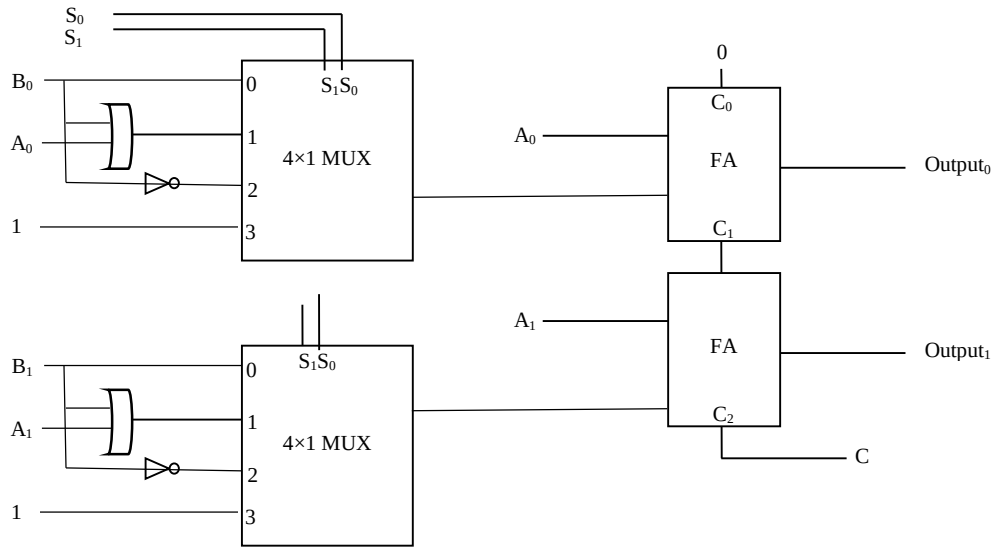
Design an arithmetic circuit for two n-bit data inputs A and B. the circuit generates the following arithmetic operations based on the selection specified, **assuming that the C_{in} of the full adder for the least significant bit is in all cases zero:**

S_1S_0	Operation
0 0	Output = A+B
0 1	Output = A+ (AvB)
1 0	Output = A+B′
1 1	Output = A−1

Draw the logic diagram for the first two stages.

[6 marks]

Solution:



Exercise 8:

a) A digital computer has a common bus system for 64 registers of 8 bits each. If the bus is constructed with multiplexers.

i. How many multiplexers are there in the bus?
ii. What size of multiplexers is needed?

b) A digital computer has a common bus system for 16 registers of 64 bits each. If the bus is constructed with multiplexers.

i. How many multiplexers are there in the bus?
ii. What size of multiplexers is needed?

[4 marks]

Solution:

a)

i. 8 multiplexers, one for each bit of the registers.

ii. 64 × 1 multiplexers.

b)

i. 64 multiplexers, one for each bit of the registers.

ii. 16 × 1 multiplexers.

Exercise 9:

a. Starting from an initial value of R = 10111001, determine the sequence of binary values in R after a circular shift-right, followed by a logical shift left, followed by a logical shift right and a circular shift right, arithmetic shift right, followed by an arithmetic shift left. State whether an overflow occurs or not. Show your work step by step.

[4 marks]

Solution:

a. R = 10111001
Circular shift-right: 11011100
Logical shift left: 10111000
Logical shift right: 01011100
Circular shift right: 00101110
Arithmetic shift right: 00010111
Arithmetic shift left: 00101110

No overflow occurred.

b. Starting from an initial value of R = 11010100, determine how could we change its value to R = 11110110 in two different ways using the applications of logical microoperations.

[2 marks]

Solution:

b. R = 11010100

First way: perform a selective set by the binary number 00100010 using or microoperation.
Second way: perform a selective complement by the binary number 00100010 using xor microoperation.