CO Project

8&16 Bit CPU Simulation

Name:

Abdelrhman Abdelftah Kassem

18010948

8-Bit CPU Simulation:

Assembly Code for the used Example:

```
LDA     5
Add     #2
Add     6
STA     7
STOP
```

Memory Contents:

00000101	Load contents of 5 into D0
00110010	Add 2 to D0
00100110	Add the contents of Location 6 to D0
01000111	Store the contents of D0 in Location 7
11100000	Exit
00001111	Location 5 has a value of 15
00000011	Location 6 has a value of 3

Expected Output is that Location 7 now has a value of 20 (15+2+3).

Edited Code:

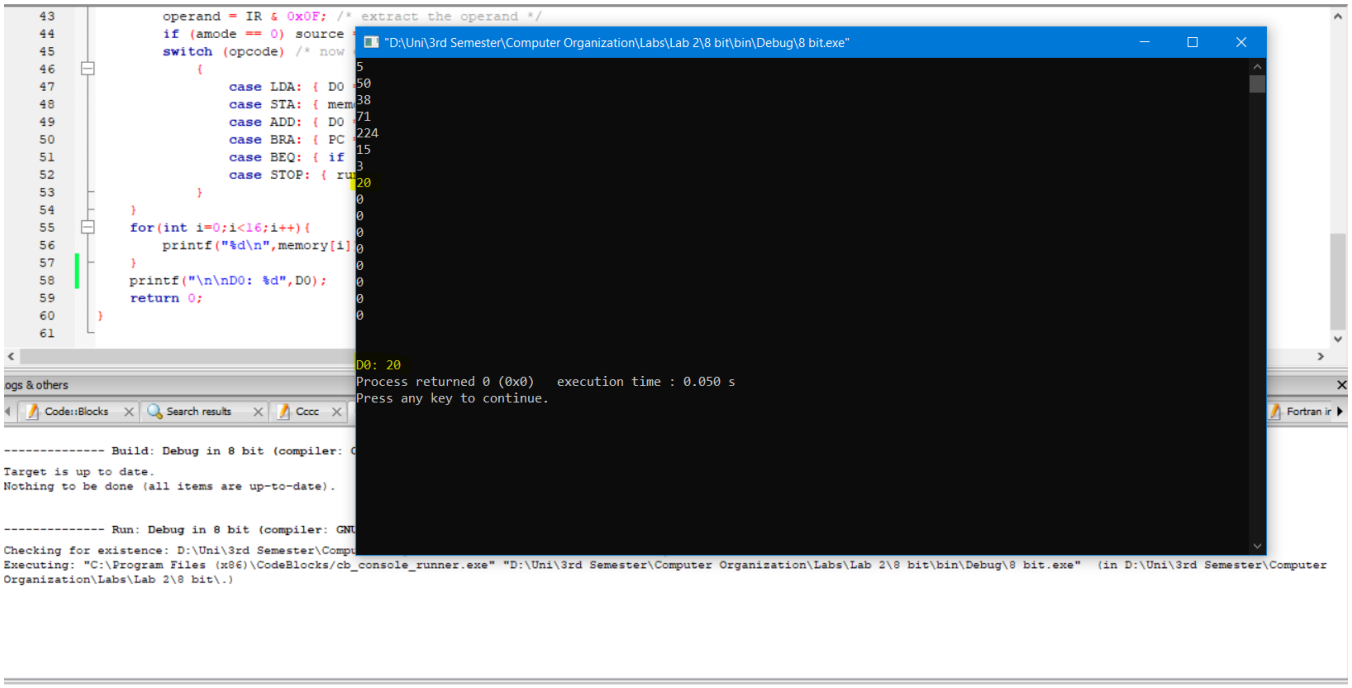
Not much was edited here just added the memory contents at the start and made a loop to print the memory contents and the value oof D0 at the end.

```
unsigned short int  PC = 0;     /*program counter*/
unsigned short int  D0 = 0;     /*data register*/
unsigned short int  MAR;        /*memory address register*/
unsigned short int  MBR;        /*memory buffer register*/
unsigned short int  IR;         /*instruction register*/
unsigned short int  operand;    /*the 8-bit operand from the IR*/
unsigned short int  source;     /*source operand*/
unsigned short int  opcode;     /*the 3-bit op-code from theIR*/
unsigned short int  amode;      /*the 1-bit addressing mode*/
unsigned short int  memory[16]={0b00000101,0b00110010,0b00100110,0b01000111,0b11100000,0b00001111,0b00000011}; /*the memory*/
unsigned short int  run = 1;    /*execute program while run is 1*/
```

```
54      }
55      for(int i=0;i<16;i++){
56          printf("%d\n",memory[i]);
57      }
58      printf("\n\nD0: %d",D0);
59      return 0;
60  }
61
```

Aside from this the code is identical to the example in the reference book.

Screenshot of the Output:



16-Bit CPU Simulation:

Assembly Code for the used Example:

```
Move 20,D0
AND 21,D0
MOVE D0,22

MOVE 20,D0
OR 21,D0
MOVE D0,23

MOVE 20,D0
XOR 21,D0
MOVE D0,24
STOP
```

This Stores contents of memory location 20 into D0 then Applies AND operation on the contents of D0 and memory location 21.

The Code is repeated for Operation OR and XOR.

Each time the result is stored in a distinct subsequent memory location to be view at the end.

Memory Contents:

00000100 – 00010100 Moves the content of memory Location 20 into D0.

10000100 – 00010101 applies AND operation on D0 and the contents of location 21.

00000000 – 00010110 Stores the content of D0 into Memory location 22.

00000100 – 00010100 Moves the content of memory Location 20 into D0.

10010100 – 00010101 applies OR operation on D0 and the contents of location 21.

00000000 – 00010111 Stores the content of D0 into Memory location 23.

00000100 – 00010100 Moves the content of memory Location 20 into D0.

10100100 – 00010101 applies OR operation on D0 and the contents of location 21.

00000000 – 00011000 Stores the content of D0 into Memory location 24.

11110000 – 00000000 Exit

10110111 Location 20 → The first operand (183).

11001011 Location 21 → The second operand (203).

After Execution.

10000011 Location 22 → The result of the AND operation (131).

11111111 Location 23 → The result of the OR operation (255)

01111100 Location 24 → The result of the XOR operation (124).

Added Code.

Defining the AND,OR,XOR operations and assigning OP-codes to them:

```
#define MOVE 0
#define ADD 1
#define SUB 2
#define BRA 3
#define CMP 4
#define BEQ 5
#define BNE 6
#define EXG 7 /*EXG exchanges the contents of two registers */
#define AND 8
#define OR  9
#define XOR 10
#define STOP 15
```

Implementing the operations:

```
        }
        case AND: {
                if (direction == 0)
                destination = D0 & source;
            else
                D0 = D0 & source;
            break;
        }
        case OR: {
                if (direction == 0)
                destination = D0 | source;
            else
                D0 = D0 | source;
            break;
        }
        case XOR: {
                if (direction == 0)
                destination = D0 ^ source;
            else
                D0 = D0 ^ source;
            break;
        }
```

Printing the required part from the memory (operands and output):

```
170         }
171         for(int i=20;i<25;i++){
172             printf("Memory Location %d : %d\n",i,memory[i]);
173         }
174     }
```

Screenshot of the Output: