EE 337: Microprocessors Laboratory (Spring 2023)

Indian Institute of Technology Bombay

Lab 2: 20 points — Date: January 12, 2023

1. [5 points] Write an assembly language program to perform modular addition and subtraction under the modulus `61H`. Use the following program as a starting point. Add your codes in the `MODADD` and `MODSUB` subroutines.

```
// -- DO NOT CHANGE ANYTHING UNTIL THE **** LINE--//
ORG 0H
LJMP MAIN
ORG 100H
MAIN:
MOV R1, #61H
CALL MODADD
CALL MODSUB
HERE: SJMP HERE
ORG 130H
// *****************

MODADD:
// ADD YOUR CODE HERE
RET
MODSUB:
// ADD YOUR CODE HERE
RET
END
```

- The inputs **x** and **y** must stored at locations `70H` and `71H` respectively.
- The sum and difference are defined as follows:

```
sum  = (x + y) mod 61H
diff = (x - y) mod 61H
```

- The sum should be stored in `72H` and the difference should be stored in `73H`.

TA Checkpoint 1

Check the following two cases:

- **x** = `34H`, **y** = `5AH`.
  Expected Results : **sum** = `2DH`, **diff** = `3BH`
- **x** = `4DH`, **y** = `09H`.
  Expected Results : **sum** = `56H`, **diff** = `44H`

2. [5 points] Write an assembly program to compute the determinant of any 2×2 matrix. Use the following program as a starting point. Add your codes in the `DET` subroutine.

```
// -- DO NOT CHANGE ANYTHING UNTIL THE **** LINE--//
ORG 0H
LJMP MAIN
ORG 100H
MAIN:
CALL DET
HERE: SJMP HERE
ORG 130H
// *****************

DET:
// ADD YOUR CODE HERE
RET
END
```

- The inputs **a**, **b**, **c** and **d** must stored at locations `60H`, `61H`, `62H` and `63H` respectively.
- The determinant of a 2 × 2 matrix is defined as follows:

  ```
  det = (a * d) - (b * c)
  ```

- The determinant will be a 16 bit number. The upper byte must be stored in `70H` and the lower byte must be stored in `71H`.

TA Checkpoint 2

Check the following two cases:

- **a** = `56H`, **b** = `12H`, **c** = `3AH`, **d** = `2CH`.
  Expected results : upper byte = `0AH`, lower byte = `B4H`
- **a** = `34H`, **b** = `E1H`, **c** = `59H`, **d** = `D0H`.
  Expected results : upper byte = `DCH`, lower byte = `07H`

3. [10 points] Bubble Sort is the simplest array sorting algorithm that works by repeatedly swapping adjacent elements if they are in the wrong order. Write an assembly language program to implement Bubble Sort algorithm to sort a given array of 8 elements in Ascending order.

   Refer to `https://www.geeksforgeeks.org/bubble-sort/` to understand Bubble Sort algorithm.

```
// -- DO NOT CHANGE ANYTHING UNTIL THE **** LINE--//
ORG 0H
LJMP MAIN
ORG 100H
MAIN:
CALL SORT
HERE: SJMP HERE
ORG 130H
// *****************

SORT:
// ADD YOUR CODE HERE
RET
END
```

   - The unsorted (input) array of elements must be stored in memory locations `60H` to `67H`.
   - The sorted elements must be stored in memory locations `70H` to `77H`.
   - To reduce the effort involved in adding multiple items in memory locations, you can use the command window in Keil.
     - Start a Keil debugging session.
     - Enter the following command in the Keil command window to load an array of 8 numbers represented in decimal format. The `I:60h` refers to indirect addressing of location `60H`. To inspect the memory, you should enter `I:0x60` in the Keil memory window.

```
E char I:60h = 14h,69h,26h,5bh,7fh,1ah,00h,05h
```

TA Checkpoint 3

Check the following 2 cases:

- Inputs samples = `14H, 69H, 26H, 5BH, 7FH, 1AH, 00H, 05H`.
  Expected output samples = `00H, 05H, 14H, 1AH, 26H, 5BH, 69H, 7FH`.
- Inputs samples = `D4H, CCH, C1H, AFH, 92H, 54H, 16H, 01H`.
  Expected output samples = `01H, 16H, 54H, 92H, AFH, C1H, CCH, D4H`.