EE337 Microprocessors Laboratory

Wadhwani Electronics Laboratory
Electrical Engineering IIT Bombay

Problem set: 2 — Date: January 15 , 2025

---

1. [5 points] Write an assembly program to add two 16-bit numbers. Use the following program as a starting point. Add your code in the `ADD16` subroutine.

```
// -- DO NOT CHANGE ANYTHING UNTIL THE **** LINE--//
ORG 0H
LJMP MAIN
ORG 100H
MAIN:
CALL ADD16
HERE: SJMP HERE
ORG 130H
// *****************

ADD16:
// ADD YOUR CODE HERE
RET
END
```

   - The first number **x** is stored at locations `70H` and `71H`, with its most significant byte (MSB) in `70H` and the least significant byte in `71H`.
   - The second number **y** is similarly stored at locations `72H` (MSB) and `73H` (LSB).
   - Since the result $\mathbf{z} = \mathbf{x} + \mathbf{y}$ can be 17 bits long, store the result in memory locations `74H`, `75H`, `76H`.
   - For $\mathbf{z} = z_{16}z_{15}z_{14}\ldots z_3z_2z_1z_0$ where z_0 is the least significant bit (LSB) and z_{16} is the most significant bit (MSB), the memory location `74H` should have $0000000z_{16}$, the memory location `75H` should have the bits $z_{15}z_{14}\ldots z_8$, and the memory location `76H` should have the bits $z_7z_6\ldots z_0$.

TA Checkpoint 1

Check the following two cases:

- $\mathbf{x} =$ `1234H`, $\mathbf{y} =$ `DCBAH`.
- $\mathbf{x} =$ `FFFFH`, $\mathbf{y} =$ `FFFFH`.

2. [5 points] Write an assembly program to swap the contents of two memory locations using XOR operation. Use the following program as a starting point. Add your codes in the `XOR_SWAP` subroutine.

   Refer to **XOR Swap Algorithm** to understand how to perform the given task.

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

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

   - The inputs **a** and **b** are stored at locations `60H` and `61H` respectively.
   - After the swap operation, location `60H` must contain the value of **b** and location `61H` must contain the value of **a**.

TA Checkpoint 2

Check the following two cases:

- **a** = `56H`, **b** = `12H`.
- **a** = `34H`, **b** = `E1H`.

3. [10 points] Write an assembly program to find the sum of all the odd numbers present from the given numbers. Use the following program as a starting point. Add your codes in the `ODD` and `ODDSUM` subroutine.

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

ODD:
// ADD YOUR CODE HERE

RET
ODDSUM:
// ADD YOUR CODE HERE

RET
END
```

- The input numbers must be stored in memory locations `60H` to `67H`.
- The result (sum) must be stored in memory location `70H`.
- To facilitate storing, we will keep the numbers small so that they will not exceed 8 bits. Hence, you may ignore the carry.
- 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,0c5h
```

TA Checkpoint 3

Check the following 2 cases:

- Inputs numbers = `04H, 02H, 06H, 05H, 03H, 01H, 00H, 10H`.
- Inputs numbers = `04H, CCH, 03H, 01H, 92H, D4H, 16H, 00H`.