Microprocessor Principles and Applications
Midterm (Hands-on Test)

Name 尤薏琇

ID F14092308

Fall 2022

The exam is 180 minutes long. The total score is **100** pts. Please read questions carefully.

Note: We may change testcases when you demonstrate your programs to us.

- ## Question 1a (15%)
  - **Description:** Please design a macro **reverse N**, which can input an 8-bit number N and output the reversed result. Please store the **result in address [0x000]**.
  - For example:

    Call instruction "**reverse 0x43**".

    0x43 is 01000011 in binary.

    So the result should be **11000010** in binary, which is the reversed result.

    [0x000] = 11000010.

10111100

5+8=10

A 2

10100010

01000101

- ## Question 1b (15%)
  - **Description**: Given an unsigned 8 bits number, please find its odd 4 bits and multiply it with its even 4 bits. Please store the **result in address [0x000].**
  - For example: 8421

    5 D

    Input 01011011

    01011011 can be regarded as 0 **1** 0 **1** 1 **0** 1 **1**

    its odd-numbered bits (the 7th, 5th, 3rd, 1st bits) is 0011, that is 0x03

    its even-numbered bits (the 6th, 4th, 2nd, 0th bits) is **1101**, that is 0x0D

    thus, the answer is 0x03 * 0x0D = 0x27

    [0x000] = 0x27

10 ⇒ input

11 ⇒ Odd 0000 1

12 ⇒ Even:

- ## Question 1c (5%)
  - **Description**: The input is two numbers (n,m). Please calculate **Combination(n,m),** which means that the number of *m*-combinations If the set has *n* elements. The value of **n will be between 3~6**, and the value of **m will be between 1~n** Please store the **result in address [0x000].**

C^5_4 $\frac{5!}{4!1!}$ $4\times5\times6$

$\frac{6!}{(3!\times(\cancel{6\times3!}))}$ $\frac{\cancel{4!}\ 4}{1!\times\cancel{3!}}$

$\frac{6!}{3!\times3!}$

- **For example:**

  Combination(n,m) = n! / m!(n-m)!

  Combination(4,1) = 4! / 1!(4-1)! = 4

  Combination(6,3) = 6! / 3!(6-3)! = 20

- **Hint:** Think about how to simplify the calculation.

C^7_3 N- $\frac{\cancel{7!}}{3!\times\cancel{4!}}$ $5\times6\times7$

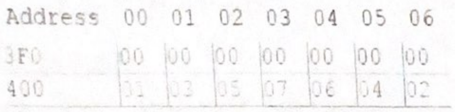
Question 2a (15%)

- **Description:** Write a macro named **LIST_INIT (n1, n2, n3, n4, n5, n6, n7)** to initialize seven 8-bit unsigned integers starting from **0x400** in memory. Then use this macro to set up one list. You are required to use at least one indirect addressing register to complete **LIST_INIT.**

- **For example:**

  Call instruction "**LIST_INIT 0x01, 0x03, 0x05, 0x07, 0x06, 0x04, 0x02**"

  the result should be like this:



Address	00	01	02	03	04	05	06
3F0	00	00	00	00	00	00	00
400	01	03	05	07	06	04	02

Question 2b (15%)

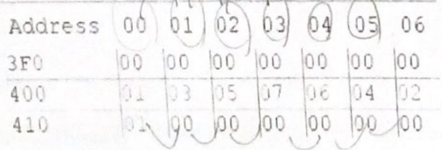
- **Description:** Implement a subroutine called **MOUNTAIN** to determine whether the input list is a mountain array. If the input list is a mountain array, load **0x01** into data memory **0x410**. Otherwise, load **0xFF** into data memory **0x410**. You are required to use at least one indirect addressing register to complete **MOUNTAIN**.

- **Hint:** Array is a mountain array if and only if there exists some i with 0 < i < arr.length - 1 such that:

  arr[0] < arr[1] < ... < arr[i - 1] < arr[i] and arr[i] > arr[i + 1] > ... > arr[arr.length - 1]

- **For example:**

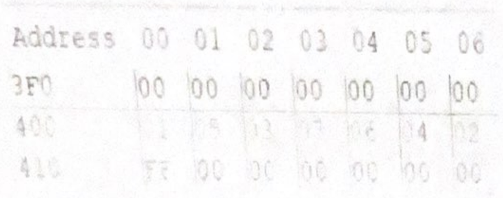
  Case 1. [0x01, 0x03, 0x05, 0x07, 0x06, 0x04, 0x02] is a mountain array.



Address	00	01	02	03	04	05	06
3F0	00	00	00	00	00	00	00
400	01	03	05	07	06	04	02
410	01	00	00	00	00	00	00

f

  Case 2. [0x01, 0x05, 0x03, 0x07, 0x06, 0x04, 0x02] is not a mountain array.



Address	00	01	02	03	04	05	06
3F0	00	00	00	00	00	00	00
400	01	05	13	07	06	04	02
410	FF	00	00	00	00	00	00

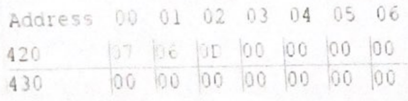
- Question 2c (5%)
  - Description: You are given a target value, please load target value into data memory **0x422**. Then you need to implement a subroutine called **TWO_SUM** to find 2 numbers in the list that their sum is equal to the target value. Please store two numbers into data memory **0x420** and **0x421**, respectively.

    You are required to use at least one indirect addressing register to complete **TWO_SUM**, and there is always **only one** pair of solution.
  - Note:
    1. The order of the two output numbers will "not" affect your score.
    2. When you implement TWO_SUM, you can change the order of elements in the list first if needed.
  - For example:

    List = [0x01, 0x03, 0x05, 0x07, 0x06, 0x04, 0x02]

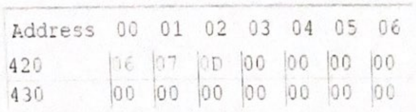
    Target value = 0x0D

    Ans: [0x420] = 0x07, [0x421] = 0x06 (or [0x420] = 0x06, [0x421] = 0x07)



Address	00	01	02	03	04	05	06
420	07	06	0D	00	00	00	00
430	00	00	00	00	00	00	00

or



Address	00	01	02	03	04	05	06
420	06	07	0D	00	00	00	00
430	00	00	00	00	00	00	00

- Question 3a (15%)
  - Description: Please implement a 16bit **BCD Adder**, and store the answer at 0x000 and 0x001
  - For example:

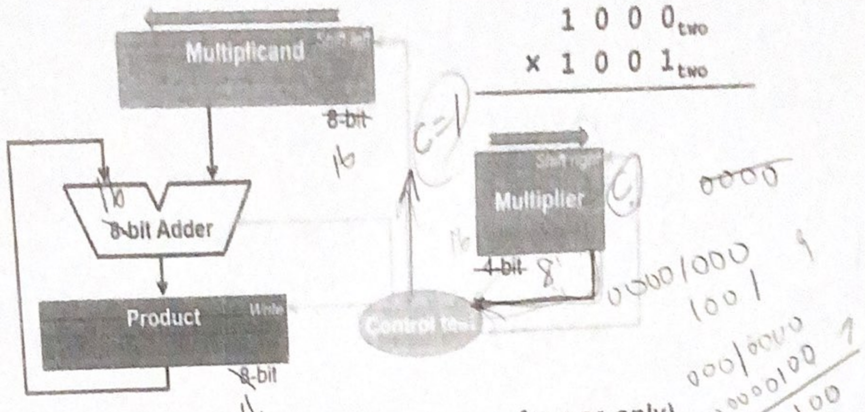
    0x1234 + 0x6666 = **0x7900** (not 0x789A)

    Then [0x000] = 0x79, [0x001] = 0x00

12 34

2345
1234
3579

- Question 3b (10%)
  - Description: Please implement a **16 bit multiplier**, the structure of the multiplier should be as **the figure shown** :



(The bit number of the figure is for reference only)

0001 0100

Please design a 16 bits multiplier on the basis of the figure shown, and store the answer at **0x000 and 0x001**.

01 11

00 01

00 11

- ○ For example:

  0x0111 x 0x0007 = 0x0777,

  Then [0x000] = 0x07, [0x001] = 0x77.

0000 0001

14 00 01 0001 0001

- ○ Note:
  1. You should implement **as the structure shown**. >0 >1
  2. **you cannot use MULWF** instruction **or continuous increase**. Otherwise you will get no point in this section.

Question 3c (5%)

- ○ Description: Please implement a program to estimate two **16 bits** contents are **palindrome** or not. Please save the estimate **result at 0x000**, if the answer is **true** then [0x000] = **0x01**, if the answer is **false** then [0x000] = **0xFF**
- ○ Hint:

  If the two 4 bits contents are 1010 and 0101, they are bilateral symmetry so they are **palindrome**.

  If the two contents are 1010 and 1010 then they are **not palindrome**.)

5F

0101 1111 ~~1111~~ 1010

5 F F A 5F FA

0101111 0101 1111

11111010 11111010