Microprocessor Principles and Applications

Midterm (Hands-on Test)

Name: 吳承恩 Fall 2023

ID: F74102145

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:** Input a number n. Convert the nth value of the Fibonacci sequence to Gray code representation and store it in [0x000]. (1 <= n <= ~~18~~ 13).
- **For example:**
  1. n=1, output 0x001.
  2. n=2, output 0x001.
- **Hint:** Here is the Gray code representation for numbers 0 to 7 in binary:

F(1) = 1
F(2) = 1
F(3) = 2
F(4) = 3

Decimal	Binary	Gray Code
0	000	**000**
1	001	**001**
2	010	**011**
3	011	**010**
4	100	**110**
5	101	**111**
6	110	**101**
7	111	**100**

Question 1b (15%)

- **Description:** Please write a subroutine called **Fib** for the Fibonacci sequence using recursion. Input a value n and store the Fibonacci sequence values from the 1st to the nth in [0x001] to [0x0n], where 1 <= n <= 13.

Question 1c (5%)

- **Description:** A fast method for calculating the Fibonacci sequence is using matrix exponentiation:

$$\begin{bmatrix} F(n) \\ F(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F(n-1) \\ F(n-2) \end{bmatrix}$$

Input a number n, then store F(n) in [0x000] for high byte, [0x001] for low byte, and F(n-1) in [0x010] for high byte, [0x011] for low byte, where 2 <= n <= 20.

- **For example:**

  1. n=2 below, we get F(2)=1, F(1)=1.

$$\begin{bmatrix} F(2) \\ F(1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Question 2a (15%)

- **Description:** Assume that you are climbing stairs. Each time you can either climb 1, 2, or 3 steps. **Given n steps of stairs to reach the top, please calculate how many distinct ways you can climb to the top**. Please show the result in Data Memory address [0x00].
- **For example:**
  1. n= 0x03, [0x00] = 0x04
  2. n= 0x04, [0x00] = 0x07
- **Note:**
  1. **You must use at least one FSR and a loop.**
  2. **$0 \leq n \leq 9$**
- **Hint:** $a_n = a_{n-1} + a_{n-2} + a_{n-3}$, $a_1 = 1$, $a_2 = 2$, $a_3 = 4$

Question 2b (15%)

- **Description:** Given 3 sorted lists, each containing three distinct integers, please merge these three lists into a single sorted list. The three sorted lists should be stored in Data Memory addresses ranging from [0x10] to [0x12], [0x20] to [0x22], and [0x100] to [0x102], respectively. The merged list should be stored in Data Memory addresses ranging from [0x00] to [0x08].
- **For example:**

  Given list: [0x010], [0x011], [0x012] = 0x01, 0x04, 0x07
  [0x020], [0x021], [0x022] = 0x02, 0x03, 0x09
  [0x100], [0x101], [0x102] = 0x05, 0x06, 0x08

  Result: [0x000] ~ [0x008] = 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09

- **Note: You must use at least one FSR.**

- **Question 2c (5%)**
  - **Description:** Implement a single turn of Quick select partitioning. Utilize the last number in the given array as the pivot and perform the first round of the Quick select partitioning process. The pseudo code is provided below. Please note that the given array contains 8 unique numbers, and the result should be stored in Data Memory address ranging from [0x00] to [0x07].
  - **For example:**

    given list = 0x01, 0x03, 0x07, 0x02, 0x09, 0x08, 0x04, 0x06

    result: [0x00] ~ [0x07] = 0x01, 0x03, 0x02, 0x04, 0x06, 0x08, 0x07, 0x09

  - **Hint:** Here is the pseudo code of Partition function:

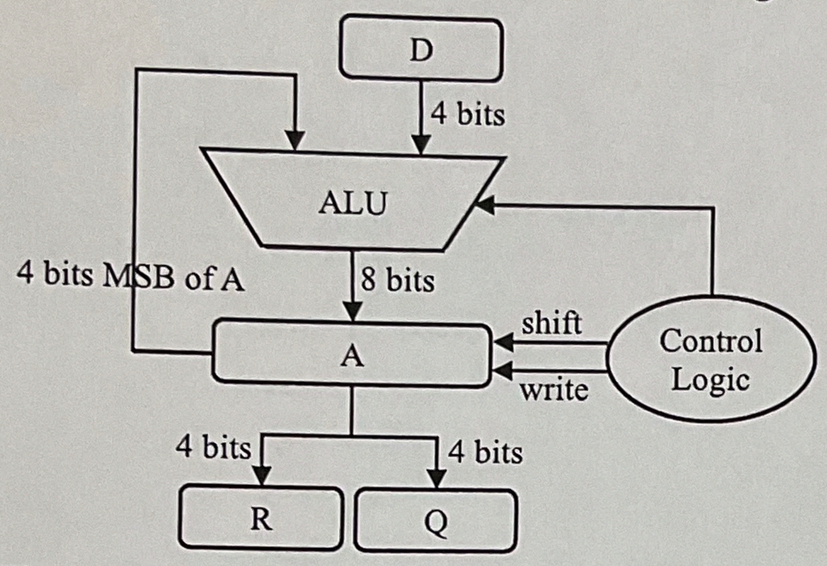
```
// Standard partition process of QuickSort().
// It considers the last element as pivot
// and moves all smaller element to left of
// it and greater elements to right
// l = 0, r = (length of arr) -1
void partition(int arr[], int l, int r)
{
    int pivot = arr[r], i = l;
    for (int j = l; j <= r - 1; j++) {
        if (arr[j] <= pivot) {
            swap(arr[i], arr[j]);
            i++;
        }
    }
    swap(arr[i], arr[r]);
}
```

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

    0x4514 – 0x1145 = 0x3369 (not 0x33CF)

    [0x000] = 0x33, [0x001] = 0x69

- **Question 3b (10%)**
  - **Description:** Please implement a **divider** which allows **8-bit dividend and 4-bit divisor,** the structure of the divider should be as the figure shown:



Please design an 8 bits divider on the basis of the figure shown, and store the **quotient and remainder** on the memory address **0x000 and 0x001**.

  - **For example:**

    0x36 ÷ 0x0B = 0x04 … 0x0A

    Result: [0x000] = 0x04, [0x001] = 0x0A

  - **Note:**
    1. You should implement as **the structure shown**.
    2. Do not use **continuous decrease**. Otherwise, you will get **no point** in this section.

- **Question 3c (5%)**
  - **Description:** Please implement a program that calculates **even parity bits for four 8-bit values**, and store the results at memory address **0x000 for bits 0, 1, 2, and 3**, respectively.
  - **For example:**

    Given number 0x28, 0x42, 0x13, 0xA7

    0x28 = b' 00101000' => 0⊕0⊕1⊕0⊕1⊕0⊕0⊕0 = 0

    0x42 = b' 01000010' => 0⊕1⊕0⊕0⊕0⊕0⊕1⊕0 = 0

    0x13 = b' 00010011' => 0⊕0⊕0⊕1⊕0⊕0⊕1⊕1 = 1

    0xA7 = b'10100111' => 1⊕0⊕1⊕0⊕0⊕1⊕1⊕1 = 1

    Result: [0x000] = b'00001100' = 0x0C