# CEC 15th - Past Year Paper Solution *2014-2015 Sem2*
## CE1006/CZ1006 – Computer Organization and Architecture

*Solver:* Jeffrey Ong

*Email:* JONG044@e.ntu.edu.sg

1. (a)

| Part | R2 | SR |
|---|---|---|
| i) | 0xABC | 0x004 |
| ii) | 0x000 | 0x002 |
| iii) | 0x9F9 | 0x004 |
| iv) | 0x991 | 0x000 |
| v) | 0x282 | 0x009 |
| vi) | 0x4C8 | 0x001 |

(b)

```
Var0 = 0;
Var1 = 10;
While (Var1 != 0) {
      If (Var1 < 5)
            Var0 ++;
      Var1 --;
}
Var0 += Var0;
```

(c) This question is tricky because PC will increment after execution of that instruction. PC will be 0x002 after execution, so [PC + 0xFFE] will be **[0x000]**.

Here comes the tricky part to find out the Hexadecimal value stored at [0x000]. Since "**ADD R3, [PC + 0xFFE]**" is the instruction saved at memory location "**0x000**" (Since PC was initially "**0x000**"), we therefore have to find out the hexadecimal value of "**ADD R3, [PC + 0xFFE]**".

(Refer to the "VIP Instruction Set Summary Sheet" for better understanding)
From the VIP Sheet, the Hex value of "**ADD R3, [PC + 0xFFE]**" will be "**0x43F**".
(Even though it is a 2 word instruction, we only need the 1st word).

Since R3 = 0x7EF, after the execution of instruction, R3 will be
0x7EF + 0x43F = **0xC2E**

2. (a)

| | | | |
|---|---|---|---|
| I1 | PSH #4 | S1 | MOVS [R3 + 0xFFF], #0 |
| I2 | PSH 0x101 | S2 | MOV [R3 + 0xFFE], [R3 + 0x004] |
| I3 | Call SubZ | S3 | MOV R1, [R3 + 0x003] |
| I4 | MOV [0x100], R0 | S4 | ADD [R3 + 0xFFF], [R1] |
| I5 | MOV SP, #0xFFF | S5 | INC R1 |
| | | S6 | DEC [R3 + 0xFFE] |
| | | S7 | JNZ Loop |

(b)

| | | |
|---|---|---|
| 0xFF7 | | |
| 0xFF8 | 0x004 | V2 |
| 0xFF9 | 0x000 | V1 |
| 0xFFA | 0xBBB | PSHM R1 |
| 0xFFB | 0xDDD | PSHM R3 |
| 0xFFC | 0x008 | Return Addr. |
| 0xFFD | 0x101 | Start Addr. |
| 0xFFE | 0x004 | Size |
| 0xFFF | | |

(c)
```
ADD SP, #2
POPM 0x00A
RET
```

3. (a)

| Device | Volatile / Non-Volatile | System / Storage Memory |
|---|---|---|
| **NAND Flash** | Non-Volatile | Storage Memory |
| **Magnetic Hard Disk** | Non-Volatile | System Memory |
| **SRAM** | Volatile | Storage Memory |

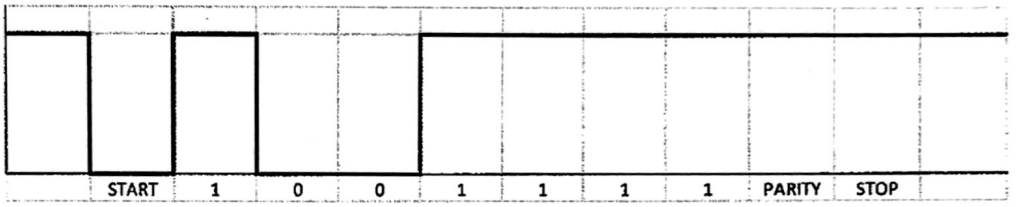
(b) Execute-in-Place (XIP) means that Program Code stored in that Flash can be executed directly without the need to transfer to RAM first.
The Parallel NOR Flash behaves like SRAM during operation.

(c) NAND Flash would be more suitable because it has lower cost per bit than NOR. Since we are required to design high capacity Flash Drives, we should choose a cheaper per bit one.

(d) (i) 19200 baud rate means it can transmit 19200 bits per second.
Since 1 character requires 10bits, therefore the maximum number of characters that can be transmitted per second is 19200 / 10 = **1920**.

**Note**: However if the 1 sec delay were to be included, then the maximum will be **1**.

(ii)




(iii) There can be 2 possible solutions to this question.
The UART receiver at the receiving end is using a RS232 configuration such that the data received will be inverted. So instead of receiving 111 1001, it receive 000 0110. The receiver baud rate is double that of the transmitter, hence sampling each bit twice. Producing the data 000 0110 (0x06).

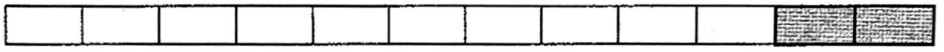
4. (a)

| Cache Mapping Scheme | Advantage | Disadvantage |
|---|---|---|
| **Direct Mapped Cache** | - Simple to implement<br>- Items can be found easily, rapidly | - Prone to cache thrashing<br>- Hit rate lowest |
| **N-Way Set Associative Cache** | - Less cache thrashing<br>- Higher hit rate for same cache size | - More complex design<br>- Longer search time |

***Thrashing**: Occurs when computer's virtual memory is in a constant state of paging

(b) According to the Principle of Locality, due to the Locality of Space (Spatial Locality), Code/Data that are close together are likely to be accessed together. Since Cache Data are fetched in blocks (contains a few words in it), the neighbouring data will will also be fetched into the Cache. The code shown is accessing data of an **array**, which means data are stored closely together (normally in the same block), hence the data can be found in Cache most of the time, causing hit rate to be high.

(c) Since the Maximum possible length is 0x3FF, which means we need 10 bits for it, leaving only **2 bits** left for the decimal. So first 10 bits are used for length, and last 2 bits will be used for precision.



To prevent the length from overflowing, we cannot add 2 length together (in case both are of max length). The only way to do it is to divide the length by 2 first, which can be done by rotating the bits right.

1. Store length of L[0] in R0, Rotate right once
2. Store length of L[1] in R1, Rotate right once
3. Add R0 and R1 together and Store in R0 (average of 2 length)
4. Store length of L[2] in R1, Rotate right once
5. Store length of L[3] in R2, Rotate right once
6. Add R1 and R2 together and Store in R1 (average of 2 length)
7. Rotate R0 and R1 right once, Add them and Store in R0 (average of 4 length)
8. Store length of L[4] in R1, Rotate right once
9. Store length of L[5] in R2, Rotate right once
10. Add R1 and R2 together and Store in R1 (average of 2 length)
11. Store length of L[6] in R2, Rotate right once
12. Store length of L[7] in R3, Rotate right once
13. Add R2 and R3 together and Store in R2 (average of 2 length)
14. Rotate R1 and R2 right once, Add them and Store in R1 (average of 4 length)
15. Rotate R0 and R1 right once, Add them and Store in R0 (average of 8 length)

**Note**: Doing this step by step is tedious but reduces the amount of precision lost.

(d) (i) R3 = 0x00B (Note that its Hexadecimal, not Decimal)

(ii) This is a Data Dependency pipeline conflict, where result of I1 is not yet available to be used by I2, causing I2 to use the previous result of R1.

To solve this issue, you can either **Stall The Pipeline** (after Decode, before Execute); or Insert **No Operation (NOP)** instructions between instructions with data dependencies.

Please pardon me for any errors as it has been a long holiday for me, memory went a little rusty!

For reporting of errors and errata, please visit pypdiscuss.appspot.com
Thank you and all the best for your exams! ☺