CEC 16th - Past Year Paper Solution *2015-2016 Sem2*
CE/CZ1006 – Computer Organisation and Architecture

Solver: Tan Hao Hao

Email Address: tanh0207@e.ntu.edu.sg

1. (a)

(i) R1: 0x7F0
SR: 0x000

(ii) R1: 0xEDD
SR: 0x004 // N flag triggered due to negative value

(iii) R1: 0x9E7
SR: 0x004 // N flag triggered due to negative value

(iv) R1: 0x216
SR: 0x009 // V flag triggered (-ve + -ve = +ve), C flag triggered

(v) R1: 0xC71
SR: 0x005 // N flag triggered (negative value), C flag triggered

(b) Since current value of PC is 0x010, assume that the program starts at address 0x010.

Address	VIP program	Remarks
0x010	START MOV R3, #2	R3: 0x002
0x012	MOV R1, #0X103	R1: 0x103
0x014	MOV R2, #2	R2: 0x002
0x016	MOV R0, [PC]	R0: 0x423
0x018	LOOP ADD R2, R3	1st loop: R2: 0x004 2nd loop: R2: 0x005 3rd loop: R2: 0x005
0x019	SUB R3, #1	1st loop: R3: 0x001 2nd loop: R3: 0x000 3rd loop: R3: 0xFFF
0x01B	JPL LOOP	1st loop: JUMP TO LOOP 2nd loop: JUMP TO LOOP 3rd loop: NO JUMP
0x01C	MOV [0X103], R3	[0X103]: 0xFFF
0x01E	FINISH RET	PC->[SP], so PC: 0x000 SP->SP+1, so SP: 0xFF4

Hence, R0: 0x423 R1: 0x103 R2: 0x005 R3: 0xFFF
SR: 0x00F SP: 0xFF4 PC: 0x000

(c) Optimized parts are labelled by *italic and underlined font* as follow:

```
START   MOVS R3, #2
        MOV R1, #0X103
        MOVS R2, #2
        MOV R0, [PC]
LOOP    ADD R2, R3
        DEC R3
        JPL LOOP
        MOV [R1], R3
FINISH  RET
```

2. (a) Pass by value – the value of the data (or variable) is passed to the subroutine
Pass by reference – the address of the variable is passed to the subroutine
VarX is passed by reference because the address 0x100 is passed to the subroutine.

(b) I2: INC SP I3: RET

(c) In binary, absolute value of a number is its 2's complement if it is negative, otherwise it would be itself. Hence the suggested code:

```
START  MOV R1, [SP+5]     // Pass parameter by reference into register
       MOV R2, [R1]       // Pass value into R2
       JPL END            // if number is positive, exit the subroutine
       EOR R2, #0xFFF     // else, invert all bits of the number
       INC R2             // then increment by 1 to achieve 2's complement
       MOV [R1], R2       // store result to the address
       POPM 15
END    RET
```

(d) A negative odd number has an MSB bit of 1 and a LSB bit of 0. Hence using Boolean algebra notation, isNegativeOdd = MSB && (LSB)', where && represents AND and ' represents NOT.

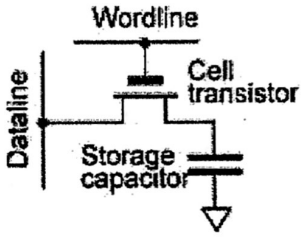
Hence the suggested code:

```
MOV R0, #VarA      // store absolute address of VarA to R0
MOV R1, [VarA]     // store value of VarA to R1
BCSR 1             // clear C flag
RAR R1             // arithmetic shift which duplicates MSB. LSB passed to C bit
MOV R2, SR         // move value of SR into R2
EOR R2, #1         // invert the LSB bit of R2
BCSR 1             // clear C flag
ROL R1             // rotate left, MSB passed to C bit
MOV R3, SR         // move value of SR into R2
AND R2, R3         // AND both values of R2 and R3
AND R2, #1         // clear all bits except LSB bit in case other bits in SR is not 0
MOV [R0], R2       // move result to the address of VarA
```

3. (a)

Cache Memory	**SRAM** because it imposes no load on the system for refreshes and is significantly faster.
System memory to be able to update and erase every byte individually	**NOR flash** because it allows random word/byte programming.
Main data storage for cloud storage services	**Magnetic HDD** because it is cheap and has large memory capacity, which is important for high demand of data storage like cloud services.

(b) Diagram of the structure of DRAM is as follow:



In a DRAM chip, each bit of memory data is stored as the presence or absence of an electric charge on a small capacitor on the chip. As time passes, the charges in the memory cells leak away, so without being refreshed the stored data would eventually be lost. To prevent this, external circuitry periodically reads each cell and rewrites it, restoring the charge on the capacitor to its original level.

(c) Two factors are:

(i) **Signal skew:**
Signal Skew is due to variation in propagation delay between signals from the same data bus caused by capacitance and resistance of the physical data line. To reduce signal skew, transfer speed is then limited.

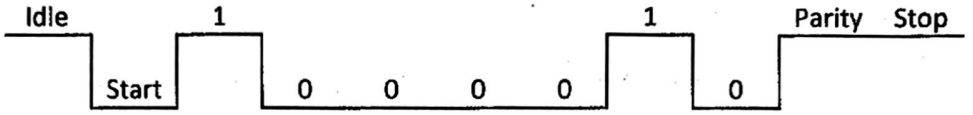
(ii) **Crosstalk:**
Crosstalk are undesired coupling of signals from one circuit to another circuit due to the close placement of the data lines in PCB routing or cabling enables the effect of electrical signal in one trace/wire to be coupled over to the other. This creates undesired interference called crosstalk.

(d) (i) The ISR is executed before a packet of data is transmitted. Hence for System A, a total of 10 + 40 = 50 instruction cycles are needed to execute ISR with CPU interrupt latency, which results in a total time of 10 * 10 microseconds = 5×10^{-4} s. Since each instruction has 10 bits, this means that the system transmits 10 bits in 5×10^{-4} s, which results in a maximum baud rate of $10 / 5 \times 10^{-4}$ s = 20000 Hz.

By the same method, we can get the maximum baud rate of System B, which is 10 / ((20 + 5) * (20 microseconds)) = 20000 Hz.

Since both values are greater than 19200, it should be a supportable baud rate.

(ii) 0x21 = 010 0001$_2$. Transferring from LSB first. Hence:



(iii) The maximum baud rate that can be choose will be 115200. This is because as delays are allowed, it allows UART to be idle, hence CPU will not be interrupted. As long as the interval between 2 consecutive messages is sufficient for CPU to process, then it doesn't matter how fast the baud rate is. Hence the maximum rate available is chosen.

4. (a)

	Advantages	**Disadvantages**
Magnetic Hard Disk Drive	Almost infinite Erasure cycles	Slower transfer rate
Solid State Drive	Higher Transfer rate	Finite number of Erasure cycles

(b) Effective access time = 0.9 X 5ns + 0.1 X (50ns + 5ns)
= 10ns

(c)

(i) Multiplying two 12-bit operands will result in a 24-bit answer, but only one 12-bit register is used to store the answer, which results in a loss of 12 bit data.

(ii) Assume that LO and HI represents the absolute address of the lower order and higher order multiplicand respectively, and R0 and R1 holds the lower order and higher order result respectively. Hence suggested code as follow:

```
      MOV AR, #12
LOOP  ROR R2
      JNC SKIP
      ADD R0, [LO]
      ADDC R1, [HI]
      SKIP BCSR 1
      RLC [LO]
      RLC [HI]
      JDAR LOOP
```

(d)

(i) In a pipeline program of **n** instructions, we need **n+3** cycles to complete since there are 4 pipeline stages.

In the code, I5, I6, I7 will be executed 10 times as loop counter AR = 10. Other instructions will be executed once. Hence we have 9 - 3 + 3*10 = 36 instructions, which results in 36 + 3 = **39** cycles.

(ii) Modifications are labelled by *italic and underlined font* as follow:

```
...                         ; I1-I4
LOOP  JDAR LOOP             ; I7
      ADD R1, [R0]          ; I5
      INC R0                ; I6
      MOV R2, R0            ; I8
      MOV R3, R1            ; I9
```

(iii) Before modifying, two cycles are wasted to calculate the branch target address and fetch the next instruction. It is shown as follow:

ADD R1, [R0]	F	D	E	S				
INC R0		F	D	E	S			
JDAR LOOP			F	D	E	S		
Next instruction						F	D	E

Hence, we modify it by swapping 2 independent instructions to the front of the branching statement as follow so no cycles will be wasted.

JDAR LOOP	F	D	E	S			
ADD R1, [R0]		F	D	E	S		
INC R0			F	D	E	S	
Next instruction				F	D	E	S

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