Solver: Mavric Tan Soon Heng

Email Address: Mavr0001@e.ntu.edu.sg

1. (a)(i) MOV R0, [0x201]
R0 = 0x999
SR = 0x004

(ii) MOV R0, [R2+0XFFD]
R0 = [0x1FF] (Not given in memory)
SR =
(I think there might be a typo in this question)

(iii) POP R0
R0 = [SP] = 0XFFF
SR = 0x000

(iv) SUB R0,R1
R0 = 0x72E
SR = 0x009

(v) RLC R0
R0 = 0x063
SR = 0x001

(vi) EOR R0, R3
R0 = 0xFC1
SR = 0x004

(b)(i)
```
while( var 1 >= var2 ){
      var1 = var 1 - var2;
}

Var3 = var 1;
```

(ii)
```
LOOP  SUB [Var1], [Var2]
      JLT Done
      JMP Loop
DONE  MOV [Var3], [Var1]
```

There will be an error if Var 1 is already lesser than Var 2.

(iii) var3 = var1% var2;

2.

(a)(i)

```
PSH [0x200]          ; Push value in memory variable VarX to stack   (I1)
PSH [0x201]          ; Push value in memory variable VarY to stack   (I2)
PSH 0x202            ; Push address of memory Ans to stack           (I3)
MOV SP, #0xFFFF      ; Remove parameters on the stack                (I4)
POPM 0x006           ; Restore used registers                        (I5)
RET                  ; Return to calling Program                     (I6)
```

b)

```
SubA  PSHM 0x006          ; Save registers R2, R1
      MOV  R1, [SP+0x006] ; Copy value of VarX to R1
      ADD  R1, [SP+0x005] ; ADD value of VarY to R1; R1= varX + varY
      MOV  R2, R1         ; Save extra copy of R1 in R2
      RCN 3               ; set rotate counter to 3
      BCSR 0x0001         ; Explicit clear carry flag
      RLC R1              ; R1 = R1 * 2^3; R1 = 8 * (varX + varY)
      ADD R2, R1          ; R2 = R1+R2; R2 = 8(varX + varY) + VarX +
                          VarY
      MOV [SP+0x004], R2  ; update variable ANS
```

It will be wrong if VarX + VarY exceeds 12bits

Alternate Answer

```
SubA  PSHM 0x006          ; Save registers R2, R1
      MOV R1, [SP+0x006]  ; R1 = VarX
      ADD R1, [SP+0x005]  ; R1 = VarX + VarY
      MOV AR, 0x008       ; Set AR to 8
Loop  ADD R1,R1           ; R1 += (VarX +Var Y)
      JDAR Loop           ; DEC AR, loop if AR != 0
      MOV [SP+0x004], R1  ; update variable ANS with R1
```

It will also be wrong if the resulting variable is larger than 12bits

c)

```
PSH PC              ; Save return address
ADD [SP], 3         ; to offset additional instructions introduced
JMP SubA            ; jump to subA
```

3.

(a)

SRAM	DRAM
Faster	Slower due to periodical refreshes
More Expensive per bit of memory	Cheaper per bit of memory
Physically larger in area per bit of memory	Smaller in area size per bit of memory

(b)

NOR	NAND
Can execute without RAM. Supports Execute in Place	Unable to execute without copying contents to RAM
Expensive per bit of memory	Cheaper per bit of memory. Suitable to use as storage

(c) NOR flash. Because it allows codes to be executed directly from it.

(d) Maximum number of UART_RX_ISR is 2500

One interrupt can only be triggered every 40 instructions (10 due to latency and 30 due to instructions length of the interrupt itself)

Hence,

Time to service 1 interrupt = (10+30) * 10 * 10^6 seconds

Amount of Interrupts per second = 1 / (10+30) * 10 * 10^6 seconds
= 2500

(ii) Maximum of packets System B can send is 3333.

One transmission requires 15 instructions (5 due to latency and 10 due to instructions length of the transmission)

Hence,

Time to transmit 1 packet = (5+10) * 10 * 10^6 seconds

Amounts of packets per second = 1 / (5+10) * 10 * 10^6 seconds
= 3333 packets

(iii) Every packet from system B triggers an interrupt in system A.
But Sys B can transmit 3333 packets while Sys A can only handle 2500 interrupts.
Hence Sys B can only transmit a max of 2500 packets to sys A in one second.

One packets consist of 1 start bit, 7 data bits, 1 parity bit, 1 stop bit, a total of 10bits.
Therefore Sys B should only use a baud rate that is lower than 2500packets * 10bits 25000bits/second. Only 19200 baud rate can be used.

Using a baud rate of 19200, only 1920 packets will be send from sys B to sys A. Since each packet only contains 7 bits, total bytes sent will be 1920 * 7 / 8 = 1680 bytes.

(iv) The current configuration does not allow 1800bytes to be sent. One way of allowing it to be met is to use the parity bit for data resulting in a (1 start bit, 8 data bits and 1 stop bit) UART configuration. The resulting system will allow 1920bytes per second

4. (a)(i) Principle of Spatial locality

If a memory address is accessed, there is a high likelihood of the nearby addresses to be accessed as well. Example, arrays

Principle of temporal locality

If a memory address is accessed, there is a high likelihood of it being accessed again. Example, loops

(b) Format of virtual address space

6 bits for page number, 10 bits for offset

Format of physical address space

2 bits for page number, 10 bits for offset

Virtual address 0x0E00 = **0000 11**10 0000 0000 (bold is page number)

page number = 3

Page number 3 is mapped to frame 1 in memory.

Physical address = **01**10 0000 0000 (bold for frame number)

= 0x600

(c) $X = -11 = 10101_2$

$Y = 11 = 01011_2$

Y should be used as it contains more consecutive ones and zeros

```
              10101
X             01011
        00000 01011 [10]
        00000 0000  [11]
        11110 101   [01]
        00010 11    [10]
        11010 1     [01]
        11100 00111
```

(d) I3 and I4 can be inserted into branch delay slots as they do not affect the branch logic

(ii) I4 cannot be inserted into branch delay as it will affect I6 which is used to compute the branch JNZ loop to see if branching is happening or not.

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