**Practice Test**

**Note: Following are some questions which you may try to solve. You want to find correct answer later on then you can use keil compiler.**

**Mid-term August 2017 may or may not have multiple choice questions.**

1. In a Little-Endian (LE) memory system, the byte at a word-aligned address is the least significant byte in the word at that address. Consider that the address 0x100 has a data 0x11 stored. Similarly, the addresses 0x101, 0x102 and 0x103 have the data 0x22, 0x33 and 0x44 stored respectively. If the register R5 = 0x100, then the contents of the register R4, after performing the operation LDR R4,[R5] (in the LE configuration), will be:
   1. 0x11223344
   2. 0x44332211
   3. 0x22114433
   4. 0x33441122
2. The register R5 has a 32-bit control information. Bit[3] of this control information corresponds to enabling and disabling of caches. The Register R5 has a value of 0xFECBCA56. If Bit[3] = 0, the caches are disabled and if Bit[3] = 1, the caches are enabled. (Note that Bit[0] corresponds to LSB and Bit[31] corresponds to MSB of R5).
   1. To enable the caches, Read-Modify-Write sequence of R4 contents is required to make Bit[3] = 1.
   2. No operation required since Bit[3] is already 1.
   3. To enable the caches, only Write sequence is sufficient to make Bit[3] = 1.
   4. To enable the caches, Write-Modify-Read sequence of R4 contents is required to make Bit[3] = 1.
3. Assume the following initial state of the registers. R1 = 0x100 and R2 = 0x200. The address 0x100 has a value of 0xAA and the address 0x200 has a value of 0xBB. Following will be the register and the memory contents after executing the instruction STM R1,{R2}. (Note: [0x100] indicates the contents of the address 0x100)
   1. R1 = 0x100, R2 = 0xAA, [0x100] = 0xAA, [0x200] = 0xBB
   2. R1 = 0xAA, R2, 0xBB, [0x100] = 0x200, [0x200] = 0x100
   3. R1 = 0x100, R2 = 0x200, [0x100] = 0x200, [0x200] = 0xBB
   4. R1 = 0x200, R2 = 0x100, [0x100] = 0xBB, [0x200] = 0x200
4. What will be the contents of R2 after the execution of the following piece of code:

LDR R1,=0xAABBCCDD

MOV R2,#0x4

AND R1,R1,#0x4

ADDNE R2,R2,#0x4

* 1. R2 = 0x4
  2. R2 = 0x8
  3. R2 = 0xAABBCCDD
  4. R2 = 0xAABBCCD4

1. The processor mode information is present in CPSR. The processor mode is currently User mode (Unprivileged mode) and needs to switch to Supervisor mode (Privileged mode). One of the following operations needs to be performed to switch from User mode to Supervisor mode.
   1. Program the mode bits in the CPSR as Supervisor mode using MSR instruction.
   2. A SWI instruction has to be executed.
   3. Reset of the processor is the only method to switch from Unprivileged mode to Privileged mode.
   4. By changing the condition code flags of the CPSR.
2. Consider the following piece of code:

MOV r8,#0x00

MSR CPSR,#0x13 ;Switch to SVC mode

MOV r8,#0xAA

MSR CPSR,#0x11 ;Switch to FIQ mode

MOV r8,#0xBB

MSR CPSR,#0x13 ;Switch to SVC mode

MOV r7,r8

The value of r7 after the execution of above piece of code is:

* 1. 0x00
  2. Unpredictable value
  3. 0xAA
  4. 0xBB

1. The link register in case of UNDEFined instruction contains the address of the UNDEFined instruction + 4. In the following piece of code, assume that the execution of the instruction UNDEF\_INSTR generates an UNDEF exception. Also, assume that the UNDEF exception vector address is 0x4.

0x100 MOV R1,#0x0

0x104 UNDEF\_INSTR

0x108 MOV R2,R1

Following is the UNDEF exception handler code:

0x4 ADD R1,R1,#0x4

0x8 SUBS R15,R14,#0x4

The value of R2 after the execution of the above piece of code (i.e.., after the execution of the main code present at 0x100, 0x104 and 0x108) is

* 1. R2 = 0x0
  2. R2 = 0x4
  3. The instruction present at 0x104 (UNDEF\_INSTR) is recursively executed and hence never reaches the instruction preset at 0x108
  4. R2 = 0x8

1. An unpipelined processor consumes 10ns to fetch, decode and execute an instruction. What is the time consumed to fetch, decode and execute the same instruction in a 5-stage pipelined processor. (Consider only one instruction for the calculation).
   1. 10ns
   2. 2ns
   3. 0.2ns
   4. 5ns
2. The minimum and maximum number that can be represented by a 4- bit signed-magnitude form and 2’s complement form respectively is:
   1. -7 to +8 and -7 to +8
   2. -8 to +8 and -8 to +8
   3. -7 to +8 and -8 to +8
   4. -7 to +7 and -8 to +7
3. Consider the following piece of code:

LDR R1,=0x100

LDR R2,=0xAABBCCDD

LDR R3,=0x11223344

STR R2,[R1]

ADD R1,R1,#0x2

LDRB R5,[R1]

What is the value of R5 after the execution of the above piece of code?

* 1. 0xAABBCC33
  2. 0xAABBCC22
  3. Depends on the Endianess of the System
  4. 0xBB

1. The Memory Management Unit (MMU) is programmed such that any address accessed from User mode generates a fault. Assume that the register R5 has an address of 0x100000. Assuming that the processor is in Supervisor mode, the following instruction is executed:

LDRT R1,[R5]

One of the following behaviors is true after the execution of the above instruction:

* 1. Fault is generated and the mode switches to User mode.
  2. Fault is generated but the mode remains in Supervisor mode.
  3. Fault is not generated but the mode switches to User mode
  4. Fault is not generated and the mode remains in Supervisor mode

1. The virtual address (VA) is mapped to the physical address (PA) using Memory Management Unit (MMU) programming. Assume, VA = 0x100000 and PA = 0x200000. The address 0x200000 has a data of 0xAA stored in it. Assume, R1 = 0x100000 and R2 = 0x200000.

The processor needs to read the contents of the address 0x20000 with MMU enabled and load it in the register R3. One of the following steps is required for performing this operation:

* 1. Perform LDR R3,[R1]
  2. Perform LDR R1,[R3]
  3. Perform LDR R3,[R2]
  4. Perform LDR R2,[R3]

1. There is a requirement to count from 1 to 100. One of the following statements is true for generating optimized assembly.
   1. for(i=1;i<=100,i++)
   2. for(i=100,i>=0;i--)
   3. Both of the above statements always disassembles the same assembly code
   4. The for loop cannot be disassembled
2. What is the value of R3 after the execution of the following sequence?

LDR R1,=0x12345678

MOV R2,#7

MOV R3,#1

ORR R3,R1,R3,LSL R2

* 1. R3 = 0x123456F8
  2. R3 = 0x12345778
  3. R3 = 0x12343678
  4. R3 = 0x123F5678

1. What is the value of R2 after the execution of the following sequence?

MOV R3,#0xBA

MOV R2,#0x10

BIC R2,R3,R2

* 1. R2 = 0xBB
  2. R2 = 0xCB
  3. R2 = 0xAA
  4. R2 = 0xCC

1. If current CPSR Mode == user and we get an IRQ interrupt, then:

* CPSR == user and SPSR \_user == IRQ
* CPSR == IRQ and SPSR\_irq == user
* CPSR == IRQ and SPSR\_irq == IRQ
* CPSR == user and SPSR\_user == IRQ

1. What will be the result of R0 after the following operation?

MOV R0,#1

LSL R0,R0,#4

1. 10 b) 0x10 c) 4 d) 0x4
2. What is the NZCV status on doing the following operation?

LDR R0,=0xFFFFFFFF

ADDS R0,R0,#1

1. 1010 b) 1001 c) 0110 d) 0101
2. Which of these will not set the V flag?
3. 0x70000000 + 0x70000000
4. 0x80000000 + 0x80000000
5. 0xA0000000 + 0xA0000000
6. 0xE0000000 + 0xE0000000
7. Which of these is an invalid stack operation instruction?
8. STMIA sp!,{r1,r2}
9. STMDB sp,{r1,r2}
10. LDMDB sp!,{r1,r2}
11. POP {r1,r2}
12. What is the difference between these 2 instruction?

ADD r0,r1,r2

ADDS r0,r1,r2

1. S variant does a signed addition
2. S variant adds r1 with r2 and subtracts the result from r0 and stores the value back in r0.
3. S variant adds r0 with r1 and subtracts the result from r2 and stores the value back in r0.
4. S variant adds r1 with r2, stores the value in r0 and updates the flags.
5. Which of these instructions will not update flags?
6. MRS b) MOVS c) CMP d) MRC
7. Which of these instructions will compile fail?
8. LSR r0,r0,#1 b) LSL r0,r0,#1 c) ASR r0,r0,#1 d) ASL r0,r0,#1
9. Which of the following statements is incorrect?
10. LDR can trigger an abort exception
11. LDC can trigger an abort exception and undefined exception
12. LDM cannot trigger an abort exception
13. MCR cannot trigger an abort exception

1. Which of the following instruction pairs is not valid?
2. BL func1 b) BL func1 c) BL func1 d) BL func1

BX LR MOV PC,LR MOVS PC,LR B LR

1. Instruction ‘POP {pc}’ is used in which of the following cases?
2. Branch to a particular location
3. Store the PC contents onto the stack
4. To trigger an undefined exception
5. Perform a loop operation