Name: _____

Student ID: _____

MIDTERM #1

Closed book except for distributed materials and one 4"x6" card with handwritten, original notes. ***No calculators or other electronic devices****. All questions are assumed to refer to the Cortex-M microprocessor unless otherwise specified. All work is to be your own -* ***show your work*** *for maximum partial credit.*

Question	Points	Score
1	4	
2	4	
3	4	
4	7	
5	6	
6	8	
7	4	
8	8	
9	10	
10	10	
11	15	
12	20	
Total		

1. (4pts) An EABI compliant function that returns a value will use _____ to return the value.
   a. **R0**
   b. R1
   c. SP
   d. LR
   e. PC

2. (4pts) Fill in the diagram below the value 0x00112233 in little endian format starting at address 0x20000000.

Address	Value
0x20000000	**33**
0x20000001	**22**
0x20000002	**11**
0x20000003	**00**

3. (4pts) List two attributes of the Cortex-M architecture architectural that are normally associated with a CISC architecture.

Instructions that take multiple cyles to complete (LDM/STM)
Instructions that use a different number of bytes (16 vs 32-bit instructions)

4. (7pts) What is the value R2 given the following code fragment and section of SRAM?

```
MOV         R0, #0x20000000
LDR         R1, [R0]            ; R1 = 0x80000001
TST         R1, #0x00000011     ; 0x80000001 & 0x00000011 != 0
LDRSHNE     R2, [R0, #4]        ; R2 = 0xFFFF8001
LDRHEQ      R2, [R0, #8]        ; NO OP
```

Address	Value
0x20000000	0x80000001
0x20000004	0x00008001
0x20000008	0xF0017ABE
0x2000000C	0x11008001

R2 : `0xFFFF8001`

5. (6pts)
Using two ARM assembly instructions, invert bits 31-27 and bits 9-5 of register R0. All other bits should remain unchanged. MOV32 is a pseudo instruction, so you cannot use it.

```
EOR     R0, R0, #0xF8000000
EOR     R0, R0, #0x000003E0
```

6. (8pts)
Write a minimal number of ARM assembly instructions that add the upper 16-bits of R0 to the lower 16-bits of R0. Place the results in R1. You are free to use any ARM assembly instructions.

```
; One solution.  Any solution with four or fewer instructions receives full credit
LSL     R1, R0, #16
LSR     R1, R1, #16
ADD     R1, R1, R0, LSR #16
```

7. (4pts)
Explain in detail why the initialization of the array found at the label BYTE_ARRAY has no effect.

```
    AREA    SRAM, READWRITE
BYTE_ARRAY      DCB  0x02
                DCB  0xFF
                DCB  0xAB
                DCB  0x00
    align
```

SRAM is volatile memory. This means the contents of the SRAM are lost when power is removed from the system. As a result, when we want to initialize SRAM, we need to write STR commands to initialize SRAM.

8. (8pts)
Without using any branch statements write the ARM assembly instructions that will set R1 to 1 if 0 ≤ R0 ≤ 100 AND R0 is even. In all other situations, set R1 to zero.

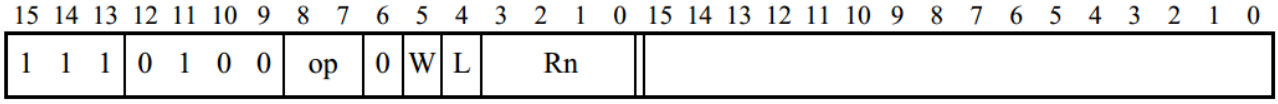
```
;If ( (0 ≤ R0 ≤ 100) && (R0 is even) )
;  R1 = 1
;Else
;  R1 = 0
```

```
MOV     R1, #1
CMP     R0, #0
MOVLT   R1, #0
CMP     R0, #100
MOVGT   R1, #0
TST     R0, #0x01
MOVNE   R1, #0
```

9. (10pts)
   a. Given the machine code encoding for the STM and LDM commands below, explain why the following two commands are equivalent.

```
LDM     R0, {R4-R8}
LDM     R0, {R8, R4, R7, R5, R6}
```

Load/store multiple



Thee lower 16 bits that can be used to specify which bits are supposed to be saved. There are no bits reserved to specifying the order in which the registers should be read/written to memory. Due to this fact, the ARM core simply writes the lowest register number to the lowest address in memory.

   b. The Rn field specifies the register used as the base address of the LDM command. Why is Rn four bits wide?
   There are 16 registers in the instruction set architecture of the Cortex-M, so we need four bits to uniquely identify which of the 16 registers is being used as the base address.
   c. List two advantages of using a LDM command to load 8 words of data as compared to 8 separate LDR commands
   A single LDM instruction will take fewer clock cycles to complete. A single LDM instruction also consumes less space in FLASH.

10. (10pts)
The function below is a non-EABI function that swaps the byte order of a WORD that has been passed into the function in R7. The value is returned in R7.

Make the modifications to the code below that will make the function EABI compliant AND minimizes the number of stack operations. Do not change which instructions are being used (AND, OR, etc). Modify only the registers being used as the parameters for an instruction.

```
ByteSwap  PROC


      PUSH {R0-R6 R4}

      MOV  R3, #0xFF

      ; Isolate Bits 7-0
      AND  R4, R3, R7 R0

      ; Isolate Bits 15-8
      AND  R5 R1, R3, R7 R0, LSR #8

      ; Isolate Bits 23-16
      AND  R6 R2, R3, R7 R0, LSR #16

      ; Isolate Bits 31-24
      AND  R7 R0, R3, R7 R0, LSR #24

      ; Perform Byte Swap
      ORR  R7 R0, R7 R0, R6 R2, LSL #8
      ORR  R7 R0, R7 R0, R5 R1, LSL #16
      ORR  R7 R0, R7 R0, R4, LSL #24

      POP {R0-R6 R4}

      BX LR


      ENDP
```

11.(15pts)
Complete the **Non-EABI callee** saved function EX1_P11 below. The function has one parameter passed in via the stack. This parameter holds the starting address of an array of 32-bit numbers. On the left, add the code requested in the comments. On the right, use section of SRAM and your code to answer the question below.

```
;*****************************
; Complete the code below
;*****************************
EX1_P11      PROC
 ; Save Registers to the Stack
 PUSH    {R0-R5}

 ; Load the address of
 ; ARRAY_LABEL passed in via
 ; the stack into R0.
  LDR     R0, [SP, #24]


 ; Load the first five WORDs
 ; of the array into R1-R5
 LDM     R0, {R1-R5}




 ; Clear bits 7-0 in each
 ; register in the range R1-R5
 BIC      R1, R1, #0xFF
 BIC      R2, R2, #0xFF
 BIC      R3, R3, #0xFF
 BIC      R4, R4, #0xFF
 BIC      R5, R5, #0xFF

 ; Write R1-R5 back to the
 ; address they were read from
 STM R0, {R1-R5}




 ; Return from the function
 POP {R0-R5}
 BX LR

    ENDP
```

Address	Value
0x20000100	0x200000CC
0x200000FC	0x200000F8
0x200000F8	0x200000FC
0x200000F4	0xCAFEF00D
0x200000F0	**0x200000C4**
0x200000EC	0x200000B0
0x200000E8	0x200000BC
0x200000E4	0x11223344
0x200000E0	0xEAB1DEAD
0x200000DC	0x200000E4
0x200000D8	0x200000F4
0x200000D4	0x200000C0
0x200000D0	0x2000FFD0
0x200000CC	0xDEADBEEF
0x200000C8	0x200000B0
0x200000C4	0x200000EC
0x200000C0	0x01234567
0x200000BC	0x200000F8
0x200000B8	0x200000DC
0x200000B4	0x200000F4
0x200000B0	0x00000000

The table above represents a section of SRAM immediately after the PUSH operation. If the stack pointer is equal to 0x200000D8, what are the values of R1-R5 when they are written back to SRAM?

R1: 0x20000000
R2: 0x20000000
R3: 0xDEADBE00
R4: 0x2000FF00
R5: 0x20000000

12. (20pts)
Write an EABI compliant routine that is used to copy memory from one memory location to another memory location. 17 points will be awarded for correctness. 3 points will be awarded for speed/efficiency.

1st parameter: Source Address
2nd parameter: Destination Address
3rd parameter: Number of words to copy

Example: Calling PROB12 in C

```
PROB12( Src, Dest, 2);    // Copy the first 2 WORDs from Src
                          // into Dest.

PROB12    PROC
     PUSH {R4-R10}

START_LDM8
     CMP  R2, #8
     BLT  START_LDM4
     LDM  R0!, {R3-R10}
     STM  R1!, {R3-R10}
     SUB  R2, R2, #8
     B    START_LDM8

START_LDM4
     CMP  R2, #4
     BLT  START_LDR
     LDM  R0!, {R3-R6}
     STM  R1!, {R3-R6}
     SUB  R2, R2, #4
     B    START_LDM4

START_LDR
     CMP  R2, #1
     BLT  END_PROB12
     LDR  R3, [R0], #4
     STR  R3, [R1], #4
     SUB  R2, R2, #1
     B    START_LDR

END_PROB12
     POP  {R4-R10}
     BX   LR
     ENDP
```