ECE 3436 Exam 2 Review Questions

Fall 2020

Note:

These examples are not the only type of questions that can appear on the exam. They are simply samples from past exams that may give you an idea of some question formats.

There is no answer key available. Contact your TAs if you have questions about your answers.

de la Rosa-Pohl

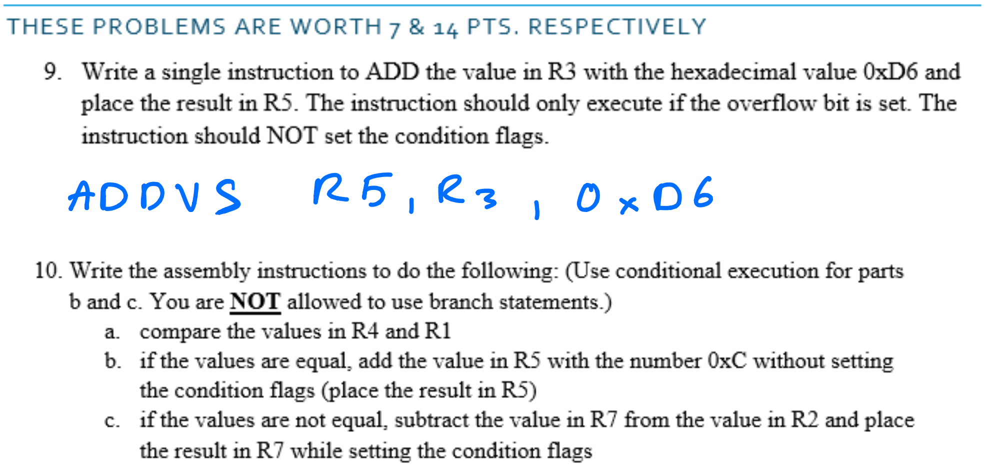
Exam 2 Major Topics

- Conditional Branching
- Programming Structures
  - including IT blocks
- GPIO
  - including bit-specific addressing
- Instruction Encoding

ORR -> Output

BIC -> Input

Conditional Execution



THESE PROBLEMS ARE WORTH 7 & 14 PTS. RESPECTIVELY

9. Write a single instruction to ADD the value in R3 with the hexadecimal value 0xD6 and place the result in R5. The instruction should only execute if the overflow bit is set. The instruction should NOT set the condition flags.

ADDVS R5, R3, 0xD6

10. Write the assembly instructions to do the following: (Use conditional execution for parts b and c. You are **NOT** allowed to use branch statements.)
    a. compare the values in R4 and R1
    b. if the values are equal, add the value in R5 with the number 0xC without setting the condition flags (place the result in R5)
    c. if the values are not equal, subtract the value in R7 from the value in R2 and place the result in R7 while setting the condition flags

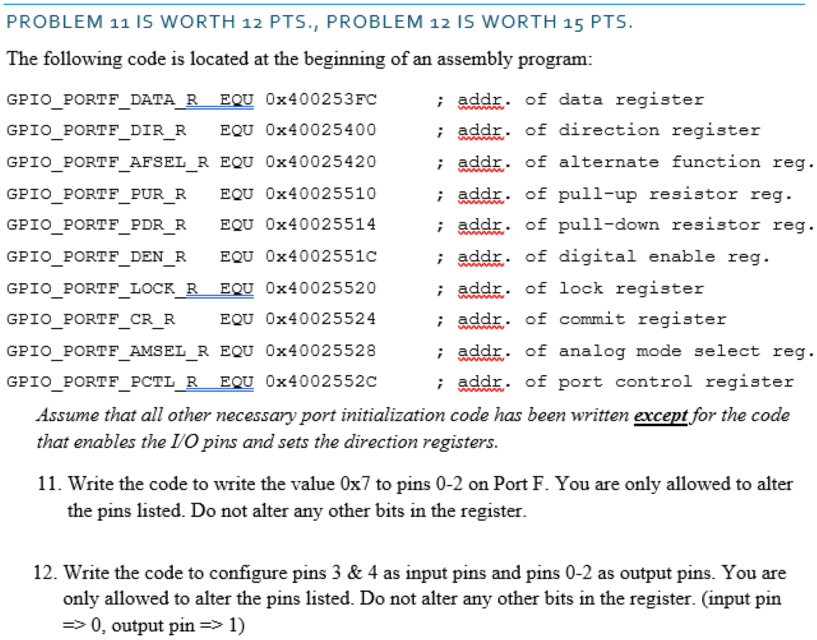
a) CMP R4, R1

b) IT EQ

ADDEQ R5, R5, 0xC

c) SUBSNE R7, R2, R7

Port Initialization



PROBLEM 11 IS WORTH 12 PTS., PROBLEM 12 IS WORTH 15 PTS.

The following code is located at the beginning of an assembly program:

```
GPIO_PORTF_DATA_R  EQU 0x400253FC     ; addr. of data register
GPIO_PORTF_DIR_R   EQU 0x40025400     ; addr. of direction register
GPIO_PORTF_AFSEL_R EQU 0x40025420     ; addr. of alternate function reg.
GPIO_PORTF_PUR_R   EQU 0x40025510     ; addr. of pull-up resistor reg.
GPIO_PORTF_PDR_R   EQU 0x40025514     ; addr. of pull-down resistor reg.
GPIO_PORTF_DEN_R   EQU 0x4002551C     ; addr. of digital enable reg.
GPIO_PORTF_LOCK_R  EQU 0x40025520     ; addr. of lock register
GPIO_PORTF_CR_R    EQU 0x40025524     ; addr. of commit register
GPIO_PORTF_AMSEL_R EQU 0x40025528     ; addr. of analog mode select reg.
GPIO_PORTF_PCTL_R  EQU 0x4002552C     ; addr. of port control register
```

Assume that all other necessary port initialization code has been written ***<u>except</u>*** *for the code that enables the I/O pins and sets the direction registers.*

11. Write the code to write the value 0x7 to pins 0-2 on Port F. You are only allowed to alter the pins listed. Do not alter any other bits in the register.

12. Write the code to configure pins 3 & 4 as input pins and pins 0-2 as output pins. You are only allowed to alter the pins listed. Do not alter any other bits in the register. (input pin => 0, output pin => 1)

0 0 0 0 0

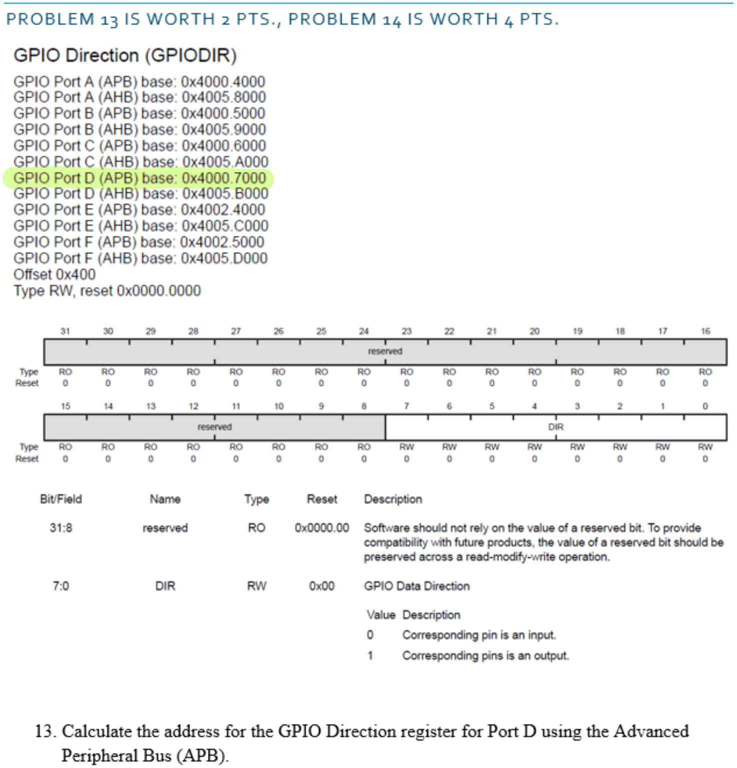
11>

```
LDR R0, =GPIO_PORTF_DATA_R
LDR R1, [R0]
ORR R1, R1, #0x7
```

12>

```
LDR R0, =GPIO_PORTF_DATA_R
LDR R1, [R0]
BIC R1, 0x18    (input for bin 3 and 4)
ORR R1, 0x7     (output for bin 0-2)
STR R0, [R1]
```

I/O Addressing



PROBLEM 13 IS WORTH 2 PTS., PROBLEM 14 IS WORTH 4 PTS.

GPIO Direction (GPIODIR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x400
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DIR							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

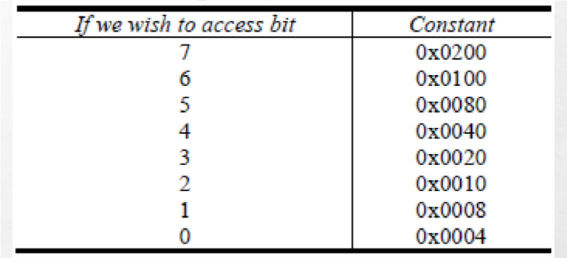
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIR	RW	0x00	GPIO Data Direction Value Description 0 Corresponding pin is an input. 1 Corresponding pins is an output.

13. Calculate the address for the GPIO Direction register for Port D using the Advanced Peripheral Bus (APB).

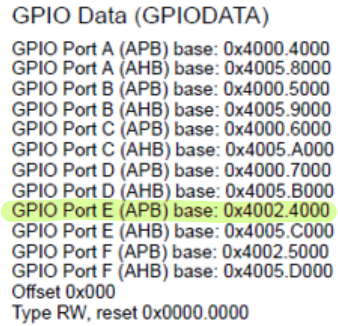
$$\text{Base + Offset} = \underline{0x400.7000 + 0x400}$$
$$= 0x400.7400$$

I/O Addressing: Bit-Specific Addressing

14. Write the address of the GPIODATA register for **Port E** using the Advanced Peripheral Bus (APB) and accessing **only** pins 0, 1, and 2. (Refer to reference sheet for bit-specific addressing table.)



If we wish to access bit	*Constant*
7	0x0200
6	0x0100
5	0x0080
4	0x0040
3	0x0020
2	0x0010
1	0x0008
0	0x0004

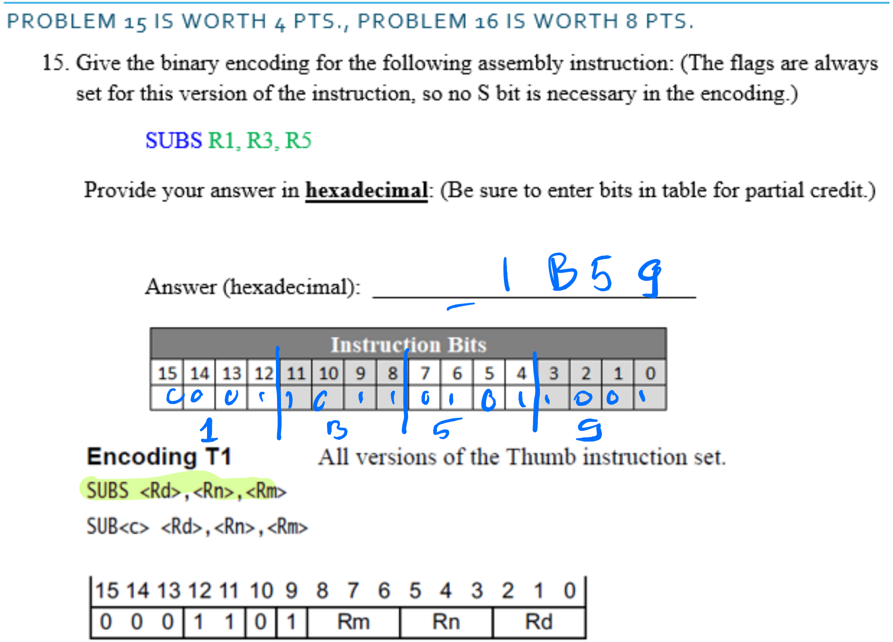


GPIO Data (GPIODATA)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x000
Type RW, reset 0x0000.0000

Answer: 0x4002.401C

Instruction Encoding



PROBLEM 15 IS WORTH 4 PTS., PROBLEM 16 IS WORTH 8 PTS.

15. Give the binary encoding for the following assembly instruction: (The flags are always set for this version of the instruction, so no S bit is necessary in the encoding.)

SUBS R1, R3, R5

Provide your answer in **hexadecimal**: (Be sure to enter bits in table for partial credit.)

Answer (hexadecimal): 1 B 5 9

Instruction Bits															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	1	1	0	1	0	1	1	0	0	1
1				B				5				9			

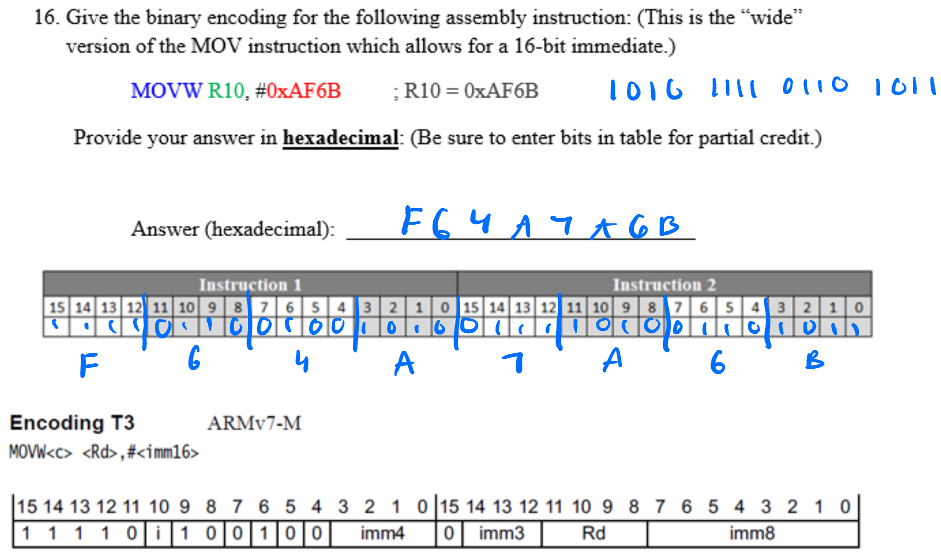
Encoding T1 All versions of the Thumb instruction set.

SUBS <Rd>,<Rn>,<Rm>

SUB<c> <Rd>,<Rn>,<Rm>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	1	Rm			Rn			Rd		

Instruction Encoding



16. Give the binary encoding for the following assembly instruction: (This is the "wide" version of the MOV instruction which allows for a 16-bit immediate.)

MOVW R10, #0xAF6B ; R10 = 0xAF6B 1010 1111 0110 1011

Provide your answer in **hexadecimal**: (Be sure to enter bits in table for partial credit.)

Answer (hexadecimal): F64A7A6B

Instruction 1																Instruction 2															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	1	0	0	1	0	0	1	0	1	0	0	1	1	1	1	0	1	0	0	1	1	0	1	0	1	1
F				6				4				A				7				A				6				B			

Encoding T3 ARMv7-M

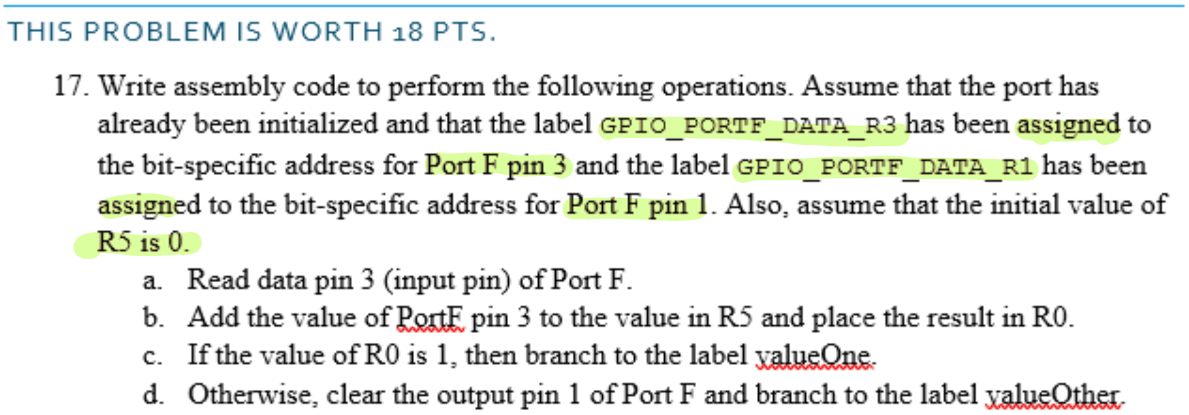
MOVW<c> <Rd>,#<imm16>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	i	1	0	0	1	0	0	imm4				0	imm3			Rd				imm8							

format for the immediate is: imm4 : i : imm3 : imm8

Assembly Programming

- GPIO Data Register Read/Write

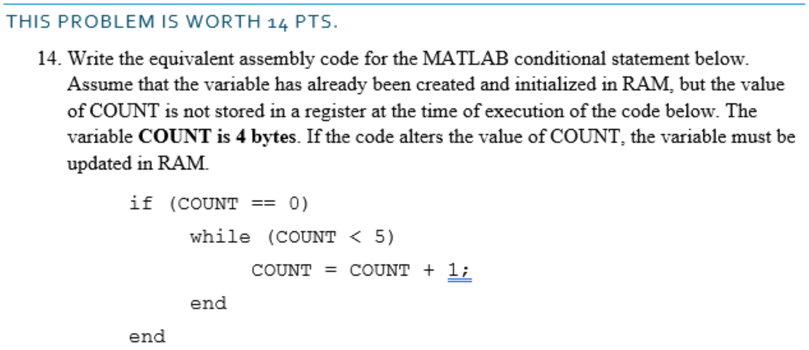


THIS PROBLEM IS WORTH 18 PTS.

17. Write assembly code to perform the following operations. Assume that the port has already been initialized and that the label `GPIO_PORTF_DATA_R3` has been assigned to the bit-specific address for Port F pin 3 and the label `GPIO_PORTF_DATA_R1` has been assigned to the bit-specific address for Port F pin 1. Also, assume that the initial value of R5 is 0.
    a. Read data pin 3 (input pin) of Port F.
    b. Add the value of PortF pin 3 to the value in R5 and place the result in R0.
    c. If the value of R0 is 1, then branch to the label valueOne.
    d. Otherwise, clear the output pin 1 of Port F and branch to the label valueOther.

```
17) a)  LDR R3, = GPIO_PORTF_DATA_R3
        LDR R2, [R3]

    b)  ADD R0, R5, R2

    c)  CMP R0, #0x1
         IT EQ
         BEQ Value One
         LDR R0, = GPIO_PORTF_DATA_R1
         LDR R1, [R0]
         BIC R1, #0x2
         BNE Value Other


Value One
     STR R0, [R3]
     NOP
Value Other
     STR R1, [R0]

    NOP
```

Programming Structures



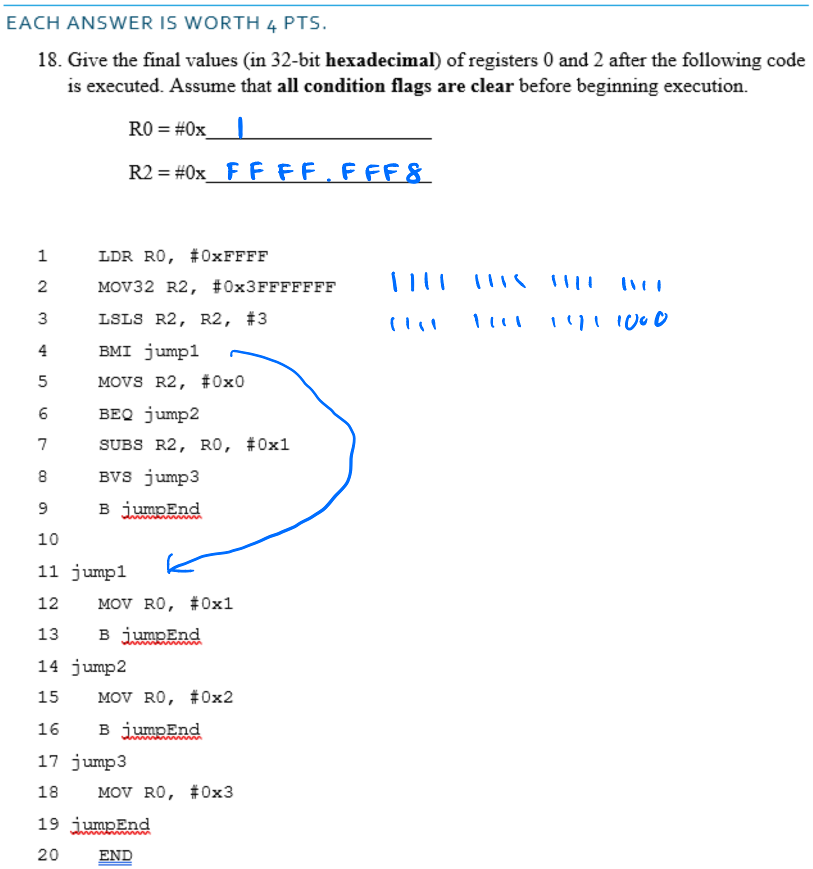
THIS PROBLEM IS WORTH 14 PTS.

14. Write the equivalent assembly code for the MATLAB conditional statement below. Assume that the variable has already been created and initialized in RAM, but the value of COUNT is not stored in a register at the time of execution of the code below. The variable **COUNT is 4 bytes**. If the code alters the value of COUNT, the variable must be updated in RAM.

```
if (COUNT == 0)
     while (COUNT < 5)
          COUNT = COUNT + 1;
     end
end
```

```
14)   LDR R0 , = COUNT
      LDR R1 , [R0]

      CMP R1 , #0

      BEQ  Loop

      BNE  END

Loop
   CMP R1 , #5

   BHS END
   BLT Increment

Increment
 ADD  R1 , #1
 STR  R0 , [R1]

 B Loop

 END
  NOP
```

Conditional Branching



EACH ANSWER IS WORTH 4 PTS.

18. Give the final values (in 32-bit **hexadecimal**) of registers 0 and 2 after the following code is executed. Assume that **all condition flags are clear** before beginning execution.

R0 = #0x 1

R2 = #0x FFFF.FFF8

```
    LDR R0, #0xFFFF
    MOV32 R2, #0x3FFFFFFF
    LSLS R2, R2, #3
    BMI jump1
    MOVS R2, #0x0
    BEQ jump2
    SUBS R2, R0, #0x1
    BVS jump3
    B jumpEnd

jump1
    MOV R0, #0x1
    B jumpEnd
jump2
    MOV R0, #0x2
    B jumpEnd
jump3
    MOV R0, #0x3
jumpEnd
    END
```

1111 1111 1111 1111

1111 1111 1111 1000

Other Possible Items

- I give you the machine code (hex or binary #) and the type of instruction (add, load, etc.) and you provide the ARM Assembly instruction.
- I tell you the general I/O configuration and you write the code for one of the steps in the GPIO configuration.
- I give you the binary or hex data in one of the GPIO configuration registers and you tell me how the I/O should work (e.g., which pins are inputs/outputs, what pins have pull-up/down resistors, how many pins are enabled, etc.)
- IT blocks
- Topics from Exam 1.