

**University of Dhaka**  
**Computer Science and Engineering**  
**3rd Year 1st Semester B.Sc. Final Examination, 2021**  
**CSE3103- Microprocessor and Microcontroller (3 Credits)**

**Time: 3 Hours**

**Total marks: 70**

Answer any five [At least two(2) from each group] out of 7 questions.

1. (a) (4 points) Briefly describe the features of the Cortex M4 memory model with appropriate diagram. What are the major address ranges?
- (b) (4 points) Explain bit banding with proper diagram. What will be the bit band alias memory location of the 11th bit of the byte offset of 0x1300 of bit band region in SRAM, where base address of bit band region (in SRAM): 0x20000000 and bit band alias region (in SRAM): 0x22000000 ?
- (c) (4 points) Describe briefly about different operating modes of ARM architecture.
- (d) (2 points) Explain the usability of Control register with different context.
2. (a) (3 points) What are the leading causes to trigger a bus fault exception in the Cortex M4 processor?
- (b) (5 points) Mention the scenario when both the Bus Fault and Hard Fault exceptions enable simultaneously. Introduce all the associated registers with their functionality which will represent the fault status in such a scenario.
- (c) (2 points) What is the purpose of Cortex M4's feature "lazy stacking"? Explain.
- (d) (4 points) Mention and describe the instructions required to call and to return from an exception. Also, list the name and functionality of different general and system registers to handle the exception.
3. (a) (4 points) Write a subroutine that checks the character in register R0 to see if it is alphabetic (upper- or lower-case). It should set the Zero flag if the character is alphabetic, and reset the flag if it is not.

Sample Problems:

		Test A	Test B	Test C
Input	R0	47 'G'	36 '6'	6A 'j'
Output	Z	FF	00	FF

- (b) (4 points) Write down the equivalent ARM assembly code of the following C code :
- ```
int main(int argc, char *argv[])
{
    int x,y;
    x = 1;
    switch(x)
    {
        case 0:
            y = 10;
```

```

break;
case 1:
y = 11;
break;
default:
y = 13;
}
return y;
}

```

- (c) (6 points) The following method returns a random number from 1 to n, where n is stored in r1.

```

# Calculate a random number
# arguments: r0 - seed (if seed is 0, get next random value)
# r1 - range (from 1 to r1). If r1 is 0 or negative, range is all ints)
#
Random:
SUB sp, sp, #8
# Save return to os on stack

STR lr, [sp, #0] Prompt For An Input
STR r4, [sp, #4]
#
MOV r3, #0
CMP r0, r3
BNE Reset
LDR r0, =seed get the seed
LDR r0, [r0, #0]
Reset:
ADD r0, r0, #137 get the next seed
EOR r0, r0, ror #13
LSR r0, r0, #1 make sure it is positive
MOV r4, r0 save the value to r4
# Get the remainder
MOV r3, #0
CMP r1, r3
BLE NoRange
BL __aeabi_idiv
MUL r1, r0, r1
SUB r4, r4, r1
NoRange:
# Save the seed to memory
LDR r0, =seed
STR r4, [r0, #0]
# Return to the OS
MOV r0, r4
LDR lr, [sp, #0]
LDR r4, [sp, #4]
ADD sp, sp, #8
MOV pc, lr
.data
seed: .word 25
#end Random

```



1. Create an array of 100 values, and populate it with 100 random numbers.
2. Using the array in previous part, find the minimum and average of all values in the array.
4. (a) (4 points) Convert the following ARM assembly code into machine language. Write the instruction in hexadecimal.
- MOV R10, #63488
  - LSL R9, R6, #7
  - STR R4, [R11, R8]
  - ASR R6, R7, R3
- (b) (2 points) Translate the following machine code into equivalent ARM assembly code.
- 0xE5902004
  - 0xE1A04638
- (c) (4 points) Describe different addressing modes used in ARM processor with proper example.
- (d) (2 points) All instructions in the ARM instruction set may be conditionally executed. Discuss the advantages and disadvantages of this feature.
- (e) (2 points) First, write down the ARM instruction to multiply the content of register r0 by nine and store the product in r7. Later, write the equivalent single instruction to perform the same operation without using any multiplication instruction.
5. GPIO and UART Configuration questions.
- (a) (6 points) Configure GPIOA PIN 0-3 for input traffic density at the crossroad and 4-7 as output to control 4-traffic lights. Assume that each pulse changes the light status, such as RED to GREEN or GREEN to YELLOW, and so on. Where MODER, SPEEDER, TYPED, and PUPDR registers have two-bit allocated for each PIN.
- (b) (4 points) Describe the steps to set a GPIO pin as input and output. Let the GPIO pin-X's speed set to 160 MHz and another GPIO pin-Y's speed set to 50 MHz. Now, to identify a real-time sensor event which pin will result best?
- (c) (4 points) A UART agent is connected to an MCU, and the MCU can read from and write values to the agent registers. MCU writes a value to the agent's register primarily to configure the device and read a register value from the agent (such as sensing the speed of a motor). However, the agent only accepts a baud rate of exactly 2MBps. Determine the mantissa and fractional values for the USARTx\_BRR register to set the baud rate. Assume the register accepts 22 bits for mantissa and only 8 bits for the fractional value.
6. Answer the following questions based on I2C deployment in the ARM-M microcontroller.
- (a) (3 points) How could an I2C slave identify the start and stop conditions of an I2C transmission? Write the steps sequentially to transfer 1MB of data using I2C.
- (b) (6 points) A temperature sensor has four registers, register-1 address 0x0AH and register-2 address 0x0BH for temperature (F/C) and pressure (p/kp) unit. Write to 0x0CH to 0x0AH set Centigrade and 0x0FH Fahrenheit accordingly. Writing 0xA0H or 0xB0H to 0x0BH sets the pressure unit to p (pascal) or Kp (kilopascal). Register-3 and Register-4 obtain the current temperature and pressure value. How can we set the temperature and pressure unit F and Kp in a single I2C frame? Similarly, read the temperature and pressure values from the sensor using an I2C frame. The address of the sensor is 0x41H. Addresses of register-3 and 4 are 0x0CH and 0x0DH respectively.



| Offset | Register    | 31                                                                  | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14       | 13     | 12    | 11     | 10  | 9     | 8   | 7     | 6        | 5     | 4      | 3    | 2       | 1       | 0    |      |
|--------|-------------|---------------------------------------------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|--------|-------|--------|-----|-------|-----|-------|----------|-------|--------|------|---------|---------|------|------|
| 0x00   | SPI_CR1     |                                                                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | BIDIMODE | BIDIOE | CRCEN | CRCEXT | DFR | RONLY | SSM | SSI   | LSBFIRST | SPE   |        | SR   |         |         |      |      |
|        | Reset value |                                                                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  | 0        | 0      | 0     | 0      | 0   | 0     | 0   | 0     | 0        | 0     | 0      | 0    | 0       | 0       | 0    |      |
| 0x04   | SPI_CR2     |                                                                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |        |       |        |     |       |     | TXEIE | RXNEIE   | ERRIE | FRF    | SSOE | TXDMAEN | RxDMAEN | CPOL | CPHA |
|        | Reset value |                                                                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |        |       |        |     |       |     | 0     | 0        | 0     | 0      | 0    | 0       | 0       | 0    | 0    |
| 0x08   | SPI_SR      |                                                                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |        |       |        |     |       | FRF | BSY   | OVR      | MODF  | CRCERR | UDR  | CHSDIE  | TXE     | RXNE |      |
|        | Reset value |                                                                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |        |       |        |     |       | 0   | 0     | 0        | 0     | 0      | 0    | 0       | 1       | 0    | 0    |
| 0x0C   | SPI_DR      | DR[15:0]                                                            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |        |       |        |     |       |     |       |          |       |        |      |         |         |      |      |
|        | Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |        |       |        |     |       |     |       |          |       |        |      |         |         |      |      |

Figure 1: SPI Essential Registers

- (c) (5 points) Many I2C devices do clock stretching to adjust the speed of the clock. However master has a timer to set the limit slave clock stretching duration. If it exceeds I2C Hang, it needs to reset. However, it is not very pleasant in real applications, and resetting may not be a suitable solution. A reset-i2c function may take the place of a hard reset. Write down the reset-i2c function with a suitable explanation.
7. Answer the following questions on SPI. Let the SPI slave contains two arrays at 0xA00H and 0xF00H to host the configuration and monitor data of crossroad traffic lights. Each configuration needs a 4-byte (uint32\_t). The registers are (i) Traffic Light 1, (ii) Traffic Light 2, (iii) Traffic Density in a north-south direction, and (iv) Traffic Density in the east-west direction. Whereas 0xF00H has eight 32-bit registers to store the last two monitoring data sets (each contains four pieces of information – lights, traffic density). Fig. 1 demonstrate the SPI essential registers.
- (a) (6 points) Write a program to configure two-MCU as master and slave SPI. Figure 1 shows the SPI registers.
- (b) (4 points) Write two functions to read monitoring data and set traffic light configuration for the Bijoy-Sarani crossroad. The address bit must clearly indicates the slave to identify read and write command. Hints: send data with the starting address.
- (c) (4 points) Explain the following figure 2 for a typical SPI communication.

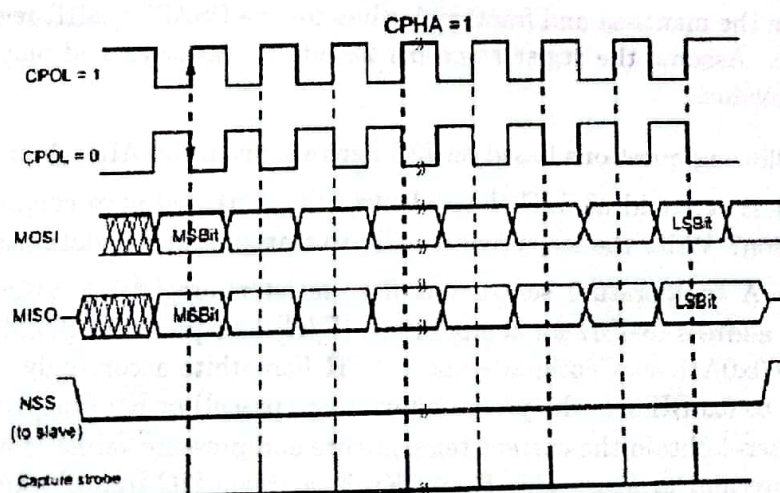


Figure 2: SPI Signaling

University of Dhaka  
Department of Computer Science and Engineering  
3rd Year 1st Semester Final Examination, 2021  
CSE-3103: Microprocessor and Microcontroller (3 Credits)  
Time: 3 hours  
Total Marks: 70

Answer any five (5) out of the following seven (7) questions.

1. (a) Convert the following ARM assembly code into machine language. Write the instructions in hexadecimal.

(i) LSR R4,R8,R6 ~~E11004638~~ **E11004638** [2]

(ii) LDR R11, [R3, #1] ~~577B0A~~ **ES93B004** [2]

(b) Translate the following machine language to the ARM assembly code.

(i) 0xE2475058 **SUB R5, R2, #8** [2]

(ii) 0xE3A0201F **MOV R2, #0x1F**

(c) Consider the following ARM assembly language snippet. The numbers to the left of each instruction indicate the instruction address.

0x000A0028 func1 MOV R4, R1 **E1A04002** [6]  
0x000A002C ADD R5, R3, R5, LSR #2 **E0835125**  
0x000A0030 SUB R4, R0, R3, ROR R4  
0x000A0034 BL func2

0x000A0038 func2 LDR R2, [R0, #1]

0x000A003C STR R2, [R1, #2]

0x000A0040 CNP R3, #0

0x000A0044 BNE else

0x000A0048 MOV PC, LR

0x000A004C else SUB R3, R3, #1

0x000A0050 B func2

(d) Translate the instruction sequence into machine code. Write the machine code instructions in hexadecimal.

(i) List the addressing mode used at each line of code.

(ii) The high-level function strcpy copies the character string src to the character string dst.

```
// C code
void strcpy(char dst[], char src[]) {
    int i = 0;
    do {
        dst[i] = src[i];
    } while (src[i++] != '\0');
}
```

Implement the strcpy function in ARM assembly code. Use R4 for i. [4]

2. (a) A subroutine is passed two integer parameters which it adds together and returns the result in R0. Assuming that the subroutine places the Link Register (LR) on the stack and uses the stack to pass the parameters, sketch the ARM code required for a simple implementation of the subroutine and draw diagrams of the resulting stack structure. [4]

(b) The following code is an implementation of Euclid's GCD algorithm. [This calculates the Greatest Common Divisor also known as the Highest Common Factor of two numbers e.g. 6 and 9 would give 3] [5]

```
gcd CMP R0, R1
BEQ end
BLT less
SUB R0, R0, R1
gcd
B gcd
less SUB R1, R1, R0
B gcd
end
```

**R0, R1**

**R0 = R1**

**R0 < R1**

(c) Design an algorithm for testing whether a given string is a palindrome. Implement your algorithm using ARM assembly code. [2]

3. (b) Describe all the special registers of cortex M4 processor with suitable diagram. [6]

(b) Explain in details the bus interfaces of cortex M4 processor. [4]

(c) Draw the cortex M4 block diagram and write down the functionality of FPU and MPU. [4]

4. (a) Explain tail chaining with a timing diagram. [4]

(b) Assume an interrupt EXTI X has the position 17 in interrupt vector table. What is the entry address for EXTI X?  $0 \times 8A$  [2]

(c) Write down the assembly instruction to disable all the interrupts with priority level equal or lower than 0x30. [2]

(d) Describe with a proper diagram the procedure to handle exception in thread mode. [6]

5. About clock and timer configuration.

(a) A microcontroller has a 16MHz externally crystal, and the clock configuration for the system uses a clock generated from the PLLCLK module of the microcontroller. The microcontroller has AHB1, APB1, and APB2 buses to connect peripherals. The peripherals connected to the AHB1 bus need an 80MHz clock. At the same time, APB2 and APB1 need clocks of 40 and 20 MHz accordingly. Determine the Prescaler values given in the following tables. [4]

| Sequence | Prescaler            | Possible values                                                                       |
|----------|----------------------|---------------------------------------------------------------------------------------|
| 1        | PLLM, PLLN, PLLP     | $2 \leq PLLM \leq 63$ , $50 \leq PLLN \leq 432$ , $PLL P \in [2, 4, 6, \text{or } 8]$ |
| 2        | System Clock Mask    | HSE, HSE, PLLCLK or PLLR                                                              |
| 3        | AHB Prescaler (HPRE) | $HPRE \in [2, 4, 8, 16, \dots, 512]$ $\phi$                                           |
| 4        | APB1 Prescaler       | $PPRE1 \in [2, 4, 8, 16]$                                                             |
|          | APB2 Prescaler       | $PPRE2 \in [2, 4, 8, 16]$                                                             |

$16 \times \frac{1}{4} \times 1600 \frac{1}{2}$

(b) The timer is an independent peripheral and is not influenced by the code running on the microprocessor, unless the code modifies its configuration. A timer can generate PWM signals as required by any application. One application lets 'AppX' need a PWM signal that comprises five pulses with duty cycles 20%, 25%, 35%, 40% and 50% accordingly. Determine the CCR1 Prescaler and ARR registers values for the up counter setting of a timer to generate the PWM signal with the given pattern. Note that an MCU changes polarity low to high or high to low whenever the CNT register matches the CCR1 register value. Assume that the input clock of the timer is 80MHz, and the generated target pulses (five altogether) take exactly five microseconds. [5]

(c) Two pushdown switches (down represents 1) are connected to pin PA5 and PA6. Four LEDs are connected to GPIO port B (PB1, PB2, PB3, and PB4). Determine the values of RCC.AHB1ENR, GPIOXMODER, GPIOXOTYPER, GPIOXOSPEEDR registers. Write a segment of code to recognize the input switches combinations, such as to light on LEDs following the conditions given in the following table. Assume that bit-0 to bit-7 must be configured to enable GPIOA to GPIOH. Starting from bit-0 to bit-31 (2-bit for each pin) of GPIOX.SPEEDR and GPIOXMODER registers represent pin 0 to 16. However, bit-0 to bit-15 needs to configure for pin-0 to pin-16 of GPIOX.OTYPER register. In addition, bit-0 to 15 for set and bit-16 to 31 for reset the 16 GPIO pins. [5]

| Input (PA5, PA6) | LED Status (PB1, PB2, PB3, PB4) |
|------------------|---------------------------------|
| 00               | on off off off                  |
| 01               | off on off off                  |
| 10               | off off on off                  |
| 11               | off off on on                   |

6. Answer the following questions based on UART deployment in the STM32F446 microcontroller.

(a) A UART slave device is connected to the MCU, and MCU can read from and write values to the device registers. MCU writing value to the register of the slave device primarily to configure the device and read a register value from the device (such as sensing the speed of the jet engine of an aircraft). However, the device only accepts a limited rate of exactly 5Mbps. Determine the transmission and fractional values for the USARTx\_BRR register to



set the baud rate. Assume that the register accepts 12-bits for mantissa and only 4-bits for the fractional value. [3]

- (b) Write a program to configure a UART communication with a baud rate 5MBps to the Jet Engine. Assume that in response to transferring 0x5C to the Jet engine, the Jet engine sends 3KBytes of data to the MCU that contains speed (in rpm), engine health information, angle to the ground height from the sea level, and GPS position. Assume that the controller uses USART6 of MCU with PC6 for Tx and PC7 for Rx. The alternate function for PC6 and 7 is AF8 (0x08). The registers for the alternate function are GPIOx\_AFR1 for '0' to '7' pins and GPIOx\_AFRH for '8' to '15' pins. RCC\_APB2ENR bits 4 and 5 for USART1 and 6 clocks enable. The USART registers are listed as follows (SR, CR1, CR2, CR3) [8]

|       |    |       |           |       |      |        |       |       |       |        |        |      |      |       |      |
|-------|----|-------|-----------|-------|------|--------|-------|-------|-------|--------|--------|------|------|-------|------|
| 31    | 30 | 29    | 28        | 27    | 26   | 25     | 24    | 23    | 22    | 21     | 20     | 19   | 18   | 17    | 16   |
|       |    |       |           |       |      |        |       |       |       |        |        |      |      |       |      |
| 15    | 14 | 13    | 12        | 11    | 10   | 9      | 8     | 7     | 6     | 5      | 4      | 3    | 2    | 1     | 0    |
|       |    |       |           |       |      | CTS    | LBD   | TXE   | TC    | RXNE   | IDLE   | ORE  | NF   | FE    | PE   |
|       |    |       |           |       |      | TC_W0  | TC_W0 | r     | TC_W0 | TC_W0  | r      | r    | r    | r     | r    |
| 31    | 30 | 29    | 28        | 27    | 26   | 25     | 24    | 23    | 22    | 21     | 20     | 19   | 18   | 17    | 16   |
|       |    |       |           |       |      |        |       |       |       |        |        |      |      |       |      |
| 15    | 14 | 13    | 12        | 11    | 10   | 9      | 8     | 7     | 6     | 5      | 4      | 3    | 2    | 1     | 0    |
| OVER8 |    | UE    | M         | WAKE  | PCE  | PS     | PEIE  | TXEIE | TCIE  | RXNEIE | IDLEIE | TE   | RE   | RWU   | SBK  |
| rw    |    | rw    | rw        | rw    | rw   | rw     | rw    | rw    | rw    | rw     | rw     | rw   | rw   | rw    | rw   |
| 31    | 30 | 29    | 28        | 27    | 26   | 25     | 24    | 23    | 22    | 21     | 20     | 19   | 18   | 17    | 16   |
|       |    |       |           |       |      |        |       |       |       |        |        |      |      |       |      |
| 15    | 14 | 13    | 12        | 11    | 10   | 9      | 8     | 7     | 6     | 5      | 4      | 3    | 2    | 1     | 0    |
|       |    | LINEN | STOP[1:0] | CLKEN | CPOL | CPHA   | LBCL  | Res.  | LBDIE | LBDL   |        |      |      |       |      |
|       |    | rw    | rw        | rw    | rw   | rw     | rw    |       | rw    | rw     |        |      |      |       |      |
| 31    | 30 | 29    | 28        | 27    | 26   | 25     | 24    | 23    | 22    | 21     | 20     | 19   | 18   | 17    | 16   |
|       |    |       |           |       |      |        |       |       |       |        |        |      |      |       |      |
| 15    | 14 | 13    | 12        | 11    | 10   | 9      | 8     | 7     | 6     | 5      | 4      | 3    | 2    | 1     | 0    |
|       |    |       |           |       |      | ONEBIT | CTSIE | CTSE  | RTSE  | DMAT   | DMAR   | SCEN | NACK | HDSEL | IRLP |
|       |    |       |           |       |      | rw     | rw    | rw    | rw    | rw     | rw     | rw   | rw   | rw    | rw   |
|       |    |       |           |       |      |        |       |       |       |        |        |      |      | IREN  | EIE  |
|       |    |       |           |       |      |        |       |       |       |        |        |      |      | rw    | rw   |

- (c) To set the USART reception interrupts, describe the bits that need to be set and check the interrupt service routine. Also, mention when the interrupt service routine for reception is activated. [3]

7. Answer the following questions on I2C?

- (a) Let an I2C master is connected to 32 slaves where each of the I2C slave maintains two addresses: (i) unique slave address and (ii) broadcast address. The broadcast address is 0xFF, and device addresses are assigned from 1 to 31. Configure both addresses for the I2C slaves. Assume that ORA1 and ORA2 contains one of the 7 and 10 bits address. The ORA1: bit-0 is ADDR0, bits-1-7 ADD[7:1] and ADD[8:9] are used for interface address, and bit-15 is the addressing mode (ADD MODE). Bit-0 of ORA2 is the EN\_DUAL, and bit-1 to 7 bits are used for the interface address. Also, draw the packet for writing 0x4C to a register (memory) identified address of 0xD5. [6]
- (b) How could a differential between start, stop, and data in I2C. If an MCU wants to read three registers values with 0xC0, 0xD0, and 0xA1 addresses, then determine the minimum number of start and stop signals that need to transfer by the master to the slave where the slave address is 0x0A. [4]
- (c) When does the slave or master stretch the clock signals (down the clock and hold)? If a master-slave communication is disrupted by clock stretching, write down the steps to release the clock to reset the I2C to the normal operational stage. For every byte of data, master, and slave exchange acknowledgment bit. Do you think the acknowledge bits play a vital role in synchronization? Explain. [4]

| Mnemonic | Opcode | Op |
|----------|--------|----|
| ADD      | 0100   | 00 |
| SUB      | 0010   | 00 |
| MOV      | 1101   | 00 |
| BL       | 1110   | 10 |
| BNE      | 0001   | 10 |
| STR      | 1110   | 01 |
| LDR      | 1110   | 01 |
| LSR      | 1110   | 00 |
| CMP      | 1010   | 00 |
| B        | 1110   | 10 |
| BLT      | 1011   | 10 |



**University of Dhaka**  
**Department of Computer Science and Engineering**  
**3rd year 1st Semester B.Sc. Final Examination-2020**  
**CSE-3103: Microprocessors and Microcontrollers**

**Total Marks: 70**

**Credits: 3**

**Time: 2 hours**

**Answer any 03 (THREE) of the following 05 (FIVE) questions**

1. a) Point out the policies adopted in ARM architecture design to address limitations of RISC architecture and to have an edge over CISC machine performance. 5.33
- b) Among the three machine language representations of ARM assembly instructions given below identify the inconsistencies (if any) and explain. [Treat each instruction individually.] 9
  - i) 1010-0000-0010-0011-1100-0101-1001-0011
  - ii) 1100-0101-0111-1000-1000-1010-1010-1010
  - iii) 0111-0101-1000-1111-0110-1010-0101-1111
- c) Convert the following three ARM assembly data processing instructions into their corresponding machine language representations. You can take help from the Appendix (Page 3) if necessary. 9
  - i) ADD r0, r1, r2
  - ii) SBCLS r3, r5, r6, LSL r0
  - iii) MOV r0, #2940 (hex value)
2. a) Suppose, r3 = 0ACB, r5 = ACB5, r6 = 0035 and r0 = 0003 (all values are in hex). 3.33  
What will be the updated value of these registers after executing the following ARM instruction:  
SUB r3, r5, r6, LSL r0  
Also provide an explanation for your answer.
- b) Explain pre-indexed and post-indexed load operation in ARM with examples. 4
- c) How does ARM handle a long range subroutine call in THUMB mode? Can this subroutine be executed in regular ARM mode? Explain. 8
- d) Draw block diagram(s) and discuss step-by-step datapath activities in 3-stage pipeline ARM organization for the following instruction: 8  
STR r3, [r0, r4, lsl #3]
3. a) What are the similarities and differences between LEA and MOV with the OFFSET directive? 3.33
- b) Explain the purpose of the following bits in the flag register of 80386: 4
  - i) IOPL ii) O iii) VM iv) T
- c) Calculate accessed memory addresses for the following instructions in real mode of 80286 given that, CS = 2000H, DS = 3000H, SS = 3000H, ES = FF00H, BX = FF04H, DI = 1020H, SI = 2030H, SP = 100H, BP = 98H. 10  
You also need to mention the updated value of the registers affected by the execution of each instruction where possible.  
Note: Treat all instructions independently.  
i) MOV DX, [BX+SI]

- ii) POP CX
- iii) MOVSB
- iv) MOV ES:[DI], CX
- v) ADD AX, [BP+2]

- d) Consider the following 80386 descriptor. Find starting and ending location indicated by it when
- i) granularity bit = 0
  - ii)granularity bit = 1

|                                 |   |   |   |    |         |                                 |                 |
|---------------------------------|---|---|---|----|---------|---------------------------------|-----------------|
| 1 0 1 0 1 1 0 0                 | G | D | L | AV | 1 1 0 0 | Access Rights                   | 0 0 0 0 1 1 1 1 |
| 0 1 0 1 0 0 0 0 1 0 1 0 1 1 1 1 |   |   |   |    |         | 0 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 |                 |

- 4

a)

Explain pipelined memory access of 80386 showing transition among different BUS states.

8

b)

Briefly describe the near and far procedure calls of the 8086 microprocessor.

8

c)

What are the benefits and disadvantages of using registers for parameter passing?

4

d)

Describe the operation of INTO instruction.

3.33
- 5

a)

Find the addressing modes for the following operations.  
i) PUSHF    ii) MOV [BP+2], CX    iii) JNZ YZ    iv) OUT 03H

8

b)

Give an example of signed overflow.

3

c)

Which flags are used for the following program control instructions?  
i) JNE    ii) SBB    iii) STC    iv) CMPSB

4

d)

What are the restrictions of MOV and XCHG instructions?

2.33

e)

Explain thread mode and handler mode of Cortex-M3. How does Cortex-M3 achieve low interrupt latency and efficient processing of late arriving interrupts?

6

| Appendix                                                              |     |                                                                                 |           |
|-----------------------------------------------------------------------|-----|---------------------------------------------------------------------------------|-----------|
| <div> <div>Opcode</div> <div>124:21)</div> <div>Mnemonic</div> </div> |     | <div> <div>Opcode</div> <div>[31:28]</div> <div>Mnemonic extension</div> </div> |           |
| 0000                                                                  | AND | 0000                                                                            | EQ        |
| 0001                                                                  | EOR | 0001                                                                            | NE        |
| 0010                                                                  | SUB | 0010                                                                            | CS/HS     |
| 0011                                                                  | RSB | 0011                                                                            | CC/LO     |
| 0100                                                                  | ADD | 0100                                                                            | Ml        |
| 0101                                                                  | ADC | 0101                                                                            | PL        |
| 0110                                                                  | SBC | 0110                                                                            | VS        |
| 0111                                                                  | RSC | 0111                                                                            | <b>VC</b> |
| 1000                                                                  | TST | 1000                                                                            | HI        |
| 1001                                                                  | TEQ | 1001                                                                            | LS        |
| 1010                                                                  | CMP | 1010                                                                            | GE        |
| 1011                                                                  | CMN | 1011                                                                            | LT        |
| 1100                                                                  | ORR | 1100                                                                            | GT        |
| 1101                                                                  | MOV | 1101                                                                            | LE-       |
| 1110                                                                  | BIC | 1110                                                                            | AL        |
| <b>1111</b>                                                           | MVN | <b>1111</b>                                                                     | NV        |
| Figure A1: ARM DPI Opcodes                                            |     | Figure A2: ARM Condition Codes                                                  |           |

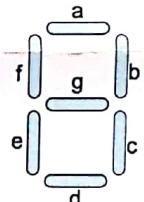


University of Dhaka  
Department of Computer Science and Engineering  
3<sup>rd</sup> Year 1<sup>st</sup> Semester B. Sc. (Hons) Final Examination, 2019  
CSE-3103: Microprocessor and Microcontroller

Total Marks: 60

Time: 3 Hours

(Answer any four (4) of the following questions)

- 1 a) Describe addressing modes of 8086 microprocessor with suitable examples. [10]  
 b) Compare memory mapped I/O and I/O mapped I/O. [3]  
 c) Describe function of ALE and  $\overline{DT}/\overline{R}$  pins of 8086. [2]
  
- 2 a) Suppose your x86 processor is equipped with peripheral controller which is connected to an 8-bit A/D converter chip. To get a sample reading from ADC, you just read at I/O address 300H. Assume that a sine wave is fed to the ADC, your job is to write an assembly language program which calculates the frequency of the sine wave. [7]  
 b) What do you mean by Assembler Directives? [3]  
 c) What is the difference between a Macro and a Procedure? [3]  
 d) Which of the followings produce the same 4 bytes? [3]  
 A1 BYTE "1234"  
 A2 BYTE 1, 2, 3, 4  
 A3 BYTE 4, 3, 2, 1  
 A4 WORD 1, 2, 3, 4  
 A5 DWORD 01020304H  
 A6 DWORD 01020304  
 A7 DWORD 04030201
  
- 3 a) A typical 7-segment display is shown below. Assume the pins (a-g) of 7-segment display are connected to data bits at I/O address 301H ( $D_0 \rightarrow a$ ,  $D_1 \rightarrow b$ , and so on). Write down an assembly language function which will take any digit between 0-9 and show that number on 7-segment display. [6]  
  
 b) What is a re-entrant procedure? [3]  
 c) Explain segment addressing in protected mode. [4]  
 d) Suppose there were no PUSH instruction. Write a sequence of two other instructions that would accomplish the same as PUSH EAX. [2]
  
- 4 a) Suppose LSB data bus of I/O port 300h is connected properly to drive a LED. Write an assembly function which can control the intensity of the LED. Intensity of LED is a numerical value from 1 to 100 supplied as a parameter of the function. [6]  
 b) Use the following data definition for this question: [4]  
 dArray DWORD 10 DUP(?)  
 dSize = (\$ - dArray)  
 byte1 BYTE 0FFh, 1, 2  
 word3 SWORD 7FFFh, 8000h

Where marked by a letter (a, b, c, d, e, f, g, h) in the following code segment, give your answer and explain your reasons. Suppose the code segment is executed sequentially from top to bottom. Note that some instructions may be illegal.

|                              |                       |
|------------------------------|-----------------------|
| mov ax, dSize                | a. ax = ?             |
| mov ax, [word3+2]            | b. ax = ?             |
| mov eax, [word3+4]           | c. eax = ?            |
| mov OFFSET byte1, 10h        | d. byte1 = ?          |
| mov ebx, OFFSET byte1        |                       |
| mov al, [ebx+3]              | e. al = ?             |
| movsx eax, byte1             | f. eax = ?            |
| mov al, 80h add al, 80h      | g. ZF, CF, SF, OF = ? |
| mov al, 00110011b test al, 2 | h. ZF, CF, SF = ?     |

- c) Translate the following C code snippet into assembly language: [3]
- ```
unsigned char data;
char count;
while(data) {
    data = data & (data - 1);
    count++;
}
printf("%d\n", count);
```

- d) Consider the following C code fragment: [2]
- ```
main(void)
{
    unsigned short us = 269;
    signed short ss = -35;

    printf("%d %d\n", us, ss);
    printf("%u %u\n", us, ss);
    printf("%hd %hd\n", us, ss);
    printf("%hu %hu\n", us, ss);
}
```

Write down the output from the program above and explain your answer.

- 5 a) What are the functions of Assembler, Linker, and Loader? [3]  
b) Consider the following C code: [4]

```
func(int i)
{
    char a[8];
    if(i==1) return 1;
    else func(i-1);
}
```

```
main(void)
{
    func(5);
}
```

Show how the stack grows before the program terminates.

- c) Write an assembly language program which can rotate a 64bit binary number (stored in RAX) to 5 places on left. You are not allowed to use the built-in rotate instruction. [4]  
d) Explain the purpose of the following flags: [4]  
i. TF  
ii. IF  
iii. OF  
iv. DF

- 6 a) Consider the following C code: [4]

```
void f(char *str)
{
    char buffer[16];
    strcpy(buffer, str);
}

void main(int argc, char **argv)
{
    f(argv[1]);
}
```

Show how stack is used to manage the local variables and function parameters.

- b) You want to exploit the stack overflow vulnerability and shift the program execution to a new address (0x55667788). What will be the input parameter to do so? [3]  
c) It is very unlikely that you get the address of some malicious code (as assumed in previous question) that you can use to exploit stack overflow. Give a more practical way to exploit stack overflow. [5]  
d) What is the benefit of using EBP to access the parameters of a function instead of ESP? [3]