Department: Computer and Systems Engineering
Academic level: 3rd **Semester:** First 2024/2025
Course code & title: Elective 2 (Embedded Systems) (CSE 4704)
Instructor: Dr. Hossam Eldin Ibrahim ALi
Total mark: 90 mark **Time allowed:** 3 hrs

Answer the following four questions:

Question 1 [18 marks]

a) Match each term or concept in Column A with its best description or corresponding example in Column B.

Column A	Column B	
1-Embedded System	B. A system that guarantees output within a defined time period.	2
2-Real-Time System	E. A specialized computer system designed for a specific application.	1
3-Engine Control Unit (ECU)	F. A device that includes I/O devices and on-chip memory, often used in embedded systems.	4
4-Microcontroller	G. A type of processor optimized for digital signal processing tasks.	5
5-Digital Signal Processor (DSP)	I. Wheel rotation sensor generating pulses.	7
6-Application-Specific Processor (ASP)	J. A type of processor with an instruction set tailored to a specific application.	6
7-Sensor Input (Bike Computer)	L. A network of distributed sensors and controllers, often using embedded systems.	10
8-Microprocessor	M. A general term for a CPU that can be used in various computing systems.	8
9-Benefit of Embedded System	N. Reduced manufacturing, operating, and maintenance costs.	9
10-Internet of Things (IoT)	O. Responsible for spark timing and fuel injection in a vehicle.	3

b) True/False Questions:

✓ 1. An IP Core is a reusable unit of logic or functionality that can be licensed to other companies.
✗ 2. A Hard IP Core is delivered as a source code file that can be modified by the user.
✗ 3. The Cortex-A series of ARM processors is designed for low-power microcontroller applications.
✓ 4. The Cortex-R series is designed for real-time applications requiring high performance and reliability.
✓ 5. ARM processors are primarily used in desktop computers and servers.
✓ 6. ARM7 has a 3-stage pipeline consisting of Fetch, Decode, and Execute stages.
✓ 7. ARM7 uses a load-store architecture, meaning that separate instructions are used for memory access (load and store) and data processing.
✗ 8. In the acronym ARM7TDMI, the "T" stands for "Timing."

Question 2 [18 marks]

a) Choose the best answer:

1. What does "Big Endian" mean in the context of memory storage?
a) Least Significant Byte is stored first [✓ b) Most Significant Byte is stored first]
c) Bytes are stored in random order

2. If the data 12345678H is stored at address 1000H in Little Endian format, what data will be at address 1000H?
a) 12H [✓ b) 78H] c) 34H

3. Which of the following is an example of a Big Endian system?
a) Intel x86 [✓ b) Motorola 68xx] c) AMD

4. Which level of cache is typically the smallest and fastest?
[✓ a) L1] b) L2 c) L3

5. What is an advantage of Big Endian representation?
a) Easier for multi-precision addition [✓ b) Easier to determine the sign of a number]
c) Easier to store data

6. In ARM7, what does the 'T' bit in the CPSR indicate?
a) Interrupt Mask [✓ b) Thumb State] c) Overflow Flag

7. Which mode in ARM7 is entered upon reset?
a) User Mode [✓ b) Supervisor Mode] c) Abort Mode

8. What is the primary advantage of using a pipeline in a processor like ARM7?
a) Simplified instruction decoding [✓ b) Faster program execution]
c) Reduced memory usage

9. Which memory section in ARM typically stores program code?
a) On-Chip Data SRAM [✓ b) On-Chip Flash ROM] c) On-Chip EEPROM
10. What is the main characteristic of cache memory?
a) It is non-volatile b) It is larger than main memory
[✓ c) It provides faster data access than main memory]

b) True or false
✓ 1. Assembly language provides direct control over hardware resources like registers and memory.
✓ 2. C language is generally considered more portable than assembly language.
✓ 3. Understanding assembly language can help in debugging and optimizing C code.
✓ 4. The linker combines object files and libraries to create an executable file.
✗ 5. The EQU directive in ARM assembly defines a new section of memory.
✓ 6. The EQU directive is used to define a constant value or address.
✗ 7 The LDR directive can only load 8-bit data into registers.
✓ 8. Data processing instructions in ARM always operate on 32-bit operands.

Question 3 [18 marks]

a) completer the following ARM7 assembly program that divides R0 by R1 using repeated subtractions.

```
; R0: Dividend                ; R1: Divisor          ; R2: Quotient (result)      ; R3: Remainder (result)
; Check for division by zero
        ………………………  ;compare divisor by #0
        BEQ division_by_zero  ; Handle division by zero appropriately
        ………………………        ; Initialize quotient to 0
        ………………………  ; Initialize remainder to dividend
        ………………………    ; Compare remainder with divisor
        ………………………    ; If remainder < divisor, division is done
        ………………………    ; Subtract divisor from remainder
        ………………………   ; Increment quotient
        ………………………    ; Loop back
        end_divide
            ; ... (Code to handle the results - R2 and R3)
        stop    B stop          ; Infinite loop to halt
        division_by_zero
            ; ... (Code to handle division by zero error)
        B stop          ; Or handle it another way appropriate to your application
```

b) For the previous program if R0 = 9 and R1 = 2, follow one iteration of the program, **<u>use the following table to show the changes in ARM registers and flags after each instruction.</u>**

instruction	Registers updates	Flags updates

Question 4 [18 marks]

a) complete the following ARM7 assembly program to calculate the sum of the squares of even numbers in an array of 100 integers. Store the sum in R8. In the same time calculate the sum of the squares of odd numbers in the same array. Store the sum in R9.

```
   AREA array_sumSQEven, CODE, READONLY
        ENTRY
        ; Define array size and address
        LDR R1, =array      ; Load address of array into R1
        ………………..        ; Load array size (100) into R2
        ; Initialize sums to 0
        MOV R8, #0         ; R8 will store the sum of even
        MOV R9, #0         ; R9 will store the sum of odd
   loop ………………..      ; Load element from array into R3, increment array pointer
         ………………..     ; square element any way, result at R4
         ………………..                   ; check if R3 even
         BNE loop2    ; if odd jump to sum odd
         ………………..       ; Add element to sum of evens
         B next
   Loop2 ………………..       ; Add element to sum of odds
   next   ………………..      ; Decrement counter, set flags
        ………………..         ; Branch to loop if counter is not zero
   stop   B stop            ; Infinite loop to halt execution
```

```
        AREA data, DATA, READWRITE
    array   DCW 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ; Example initial values (first 11)
        ; ... (Space for the remaining 64 elements)
```

b) For the previous program follow two iterations of the program. **use the following table to show the changes in ARM registers and flags after each instruction.**

instruction	Registers updates	Flags updates

Question 5 [18 marks]

a) Create ARM7 embedded C program for a 2nd order difference equation without using past y values (FIR filter). **Difference equation:** y[n] = a0*x[n] + a1*x[n-1] + a2*x[n-2],
you will read x[n] from ARM7 ADC0.1 and output the y[n] to ARM7 DAC (AOUT at P0.25).

```
#include <stdint.h>
// Filter coefficients
#define A0 0.25f
#define A1 0.5f
#define A2 0.25f
int ADC_read(){
        uint32_t result;
        AD0CR = AD0CR | (1<<24); /* Start Conversion */
        while ( !(AD0DR1 & 0x80000000) ); /* Wait till DONE */
        result = ……………………….  // read ADC data register
        ……………………….  //align ADC reading
        ……………………….  //mask 10 bits ADC reading
        return result;
}
Void DAC_output(uint16_t value){……………………….  //convert "value" to analog  }

int main(void) {
 // Initialize ADC and DAC
        PINSEL1 = 0x01000000; /* P0.28 as AD0.1 */
        AD0CR = 0x00200402; /* ADC operational, 10-bits, 11 clocks for conversion */
        PINSEL1 |= 0x00080000;      /* P0.25 as DAC output */
        IO0DIR = ( IO0DIR & 0xFFFFF0FF );
 // Variables to store past input samples
 uint16_t x_n, x_n_minus_1 = 0, x_n_minus_2 = 0;
 // Variable to store the output
 float y_n;
 while (1) {
  ………………………. // 1. Read input from ADC
  ……………………….// 2. Calculate output using the difference equation
  // 3. Output to DAC (scaling and limiting as needed)
  // Assuming DAC expects 10-bit value (0-1023), scale and limit accordingly
  uint16_t dac_output = (uint16_t)(y_n);
  if (dac_output > 1023) {     dac_output = 1023;   }
  else if (dac_output < 0) {     dac_output = 0;   }
  ……………………….  //convert dac_output to analog
  // 4. Update past input samples
  x_n_minus_2 = x_n_minus_1;
  ……………………….  }
 return 0;          }
```

b) Choose the best answer:

1. What is the resolution of the ADCs in the LPC2148?
a) 8-bit   b) 10-bit   c) 12-bit   d) 16-bit

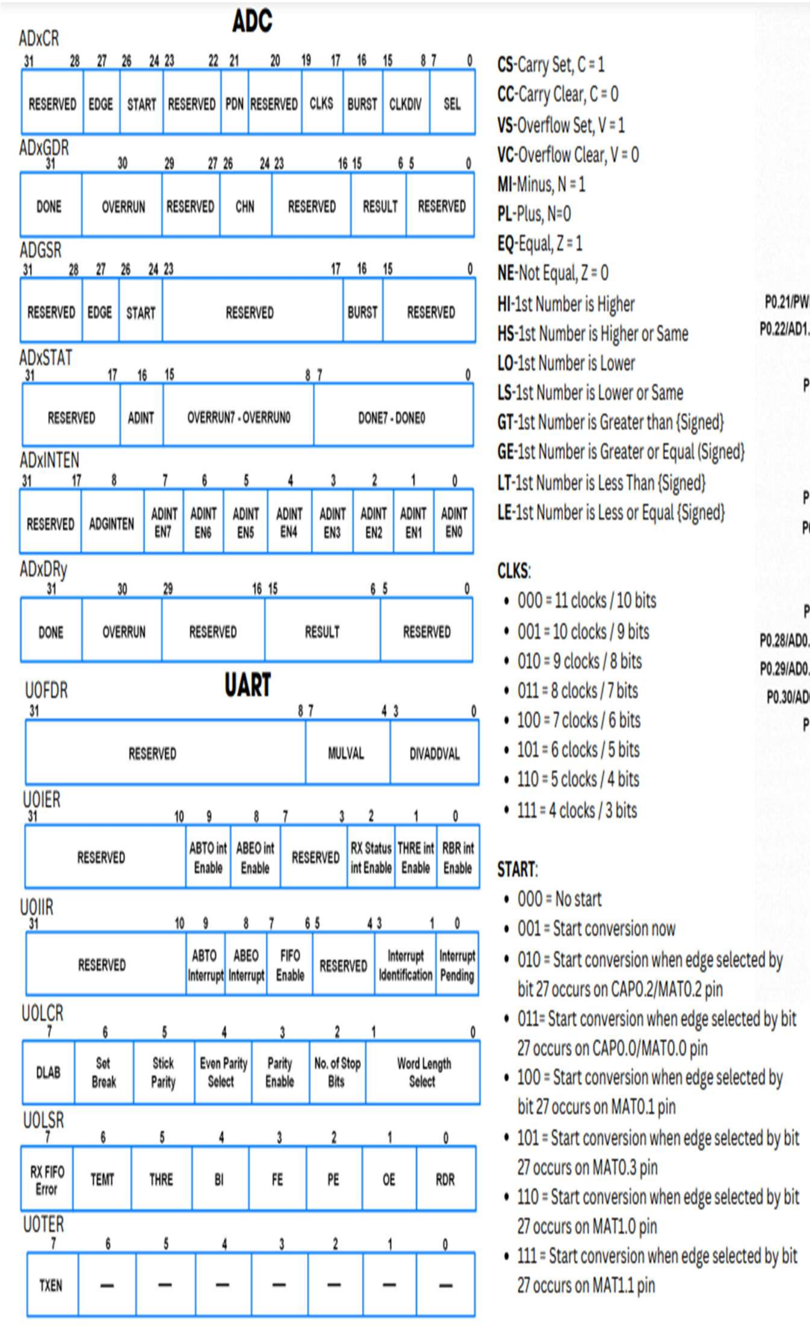
2. Which conversion technique do the LPC2148 ADCs use?
a) Flash conversion   b) Sigma-delta conversion
c) Successive approximation   d) Dual-slope integration

3. What is the maximum recommended clock frequency for the LPC2148 ADCs?
a) 1 MHz   b) 4.5 MHz   c) 10 MHz   d) 60 MHz

4. In AD0CR, what do the CLKDIV bits (15:8) determine?
a) The ADC channel to be used   b) The clock division factor for the ADC clock
c) The voltage reference selection   d) The start of conversion trigger

5. What does the BURST bit (bit 16) in AD0CR control?
a) Enables/disables the ADC b) Selects between single and repeated conversion modes
c) Sets the ADC resolution d) Starts the conversion process
6. What is the purpose of the DONE bit (bit 31) in AD0GDR?
a) Indicates if the ADC is powered on b) Indicates if the conversion is complete
c) Indicates an overrun error d) Indicates the selected ADC channel
7. In Burst mode, what does the OVERRUN bit (bit 30) in AD0DRx signify?
a) The ADC result is outside the valid range c) The ADC clock is too fast
b) One or more previous conversions were lost before the current result was stored
d) The selected channel is invalid
8. What does the AD0INTEN register control?
a) The ADC clock frequency b) Which ADC channels can generate interrupts
c) The start of conversion trigger d) The ADC resolution
9. If the 10-bit ADC result is 1023, what is the approximate input voltage (assuming VREF = 3.3V)?
a) 0V b) 1.65V c) 3.3V d) 5V
10. What is the resolution of the DAC in the LPC2148?
a) 8-bit b) 10-bit c) 12-bit d) 16-bit

With our best wishes



ADC

ADxCR

31 28	27	26 24	23 22	21	20	19 17	16	15 8	7 0
RESERVED	EDGE	START	RESERVED	PDN	RESERVED	CLKS	BURST	CLKDIV	SEL

ADxGDR

31	30	29 27	26 24	23 16	15 6	5 0
DONE	OVERRUN	RESERVED	CHN	RESERVED	RESULT	RESERVED

ADGSR

31 28	27	26 24	23 17	16	15 0
RESERVED	EDGE	START	RESERVED	BURST	RESERVED

ADxSTAT

31 17	16	15 8	7 0
RESERVED	ADINT	OVERRUN7 - OVERRUN0	DONE7 - DONE0

ADxINTEN

31 17	8	7	6	5	4	3	2	1	0
RESERVED	ADGINTEN	ADINT EN7	ADINT EN6	ADINT EN5	ADINT EN4	ADINT EN3	ADINT EN2	ADINT EN1	ADINT EN0

ADxDRy

31	30	29 16	15 6	5 0
DONE	OVERRUN	RESERVED	RESULT	RESERVED

UART

UOFDR

31 8	7 4	3 0
RESERVED	MULVAL	DIVADDVAL

UOIER

31 10	9	8	7 3	2	1	0
RESERVED	ABTO int Enable	ABEO int Enable	RESERVED	RX Status int Enable	THRE int Enable	RBR int Enable

UOIIR

31 10	9	8	7 6	5 4	3 1	0
RESERVED	ABTO Interrupt	ABEO Interrupt	FIFO Enable	RESERVED	Interrupt Identification	Interrupt Pending

UOLCR

7	6	5	4	3	2	1 0
DLAB	Set Break	Stick Parity	Even Parity Select	Parity Enable	No. of Stop Bits	Word Length Select

UOLSR

7	6	5	4	3	2	1	0
RX FIFO Error	TEMT	THRE	BI	FE	PE	OE	RDR

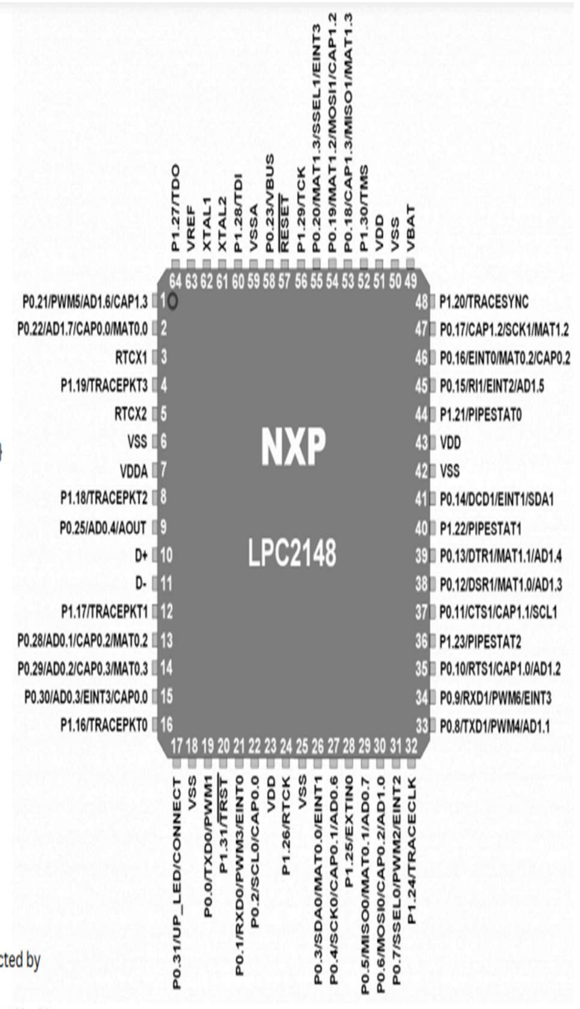
UOTER

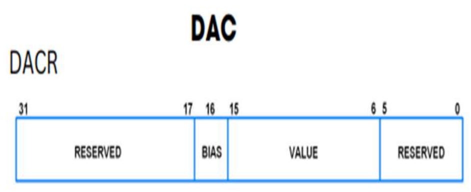
7	6	5	4	3	2	1	0
TXEN	—	—	—	—	—	—	—

CS-Carry Set, C = 1
CC-Carry Clear, C = 0
VS-Overflow Set, V = 1
VC-Overflow Clear, V = 0
MI-Minus, N = 1
PL-Plus, N=0
EQ-Equal, Z = 1
NE-Not Equal, Z = 0
HI-1st Number is Higher
HS-1st Number is Higher or Same
LO-1st Number is Lower
LS-1st Number is Lower or Same
GT-1st Number is Greater than {Signed}
GE-1st Number is Greater or Equal {Signed}
LT-1st Number is Less Than {Signed}
LE-1st Number is Less or Equal {Signed}

CLKS:
- 000 = 11 clocks / 10 bits
- 001 = 10 clocks / 9 bits
- 010 = 9 clocks / 8 bits
- 011 = 8 clocks / 7 bits
- 100 = 7 clocks / 6 bits
- 101 = 6 clocks / 5 bits
- 110 = 5 clocks / 4 bits
- 111 = 4 clocks / 3 bits

START:
- 000 = No start
- 001 = Start conversion now
- 010 = Start conversion when edge selected by bit 27 occurs on CAP0.2/MAT0.2 pin
- 011= Start conversion when edge selected by bit 27 occurs on CAP0.0/MAT0.0 pin
- 100 = Start conversion when edge selected by bit 27 occurs on MAT0.1 pin
- 101 = Start conversion when edge selected by bit 27 occurs on MAT0.3 pin
- 110 = Start conversion when edge selected by bit 27 occurs on MAT1.0 pin
- 111 = Start conversion when edge selected by bit 27 occurs on MAT1.1 pin





DAC

DACR

31 17	16	15 6	5 0
RESERVED	BIAS	VALUE	RESERVED