الاســم:

Computer and systems Engineering
Faculty of Engineering
Helwan university
Elective 2 (Embedded systems)

Midterm exam (Nov. 2024)

Question 1: introduction to embedded systems (12 marks)

a) True or false

1. Embedded systems always require a complex, high-power microcontroller. (✗)
2. A key benefit of embedded systems is the potential for lower costs due to the use of less expensive components and streamlined manufacturing. (✓)
3. A bike computer's primary sensor detects wheel movement using GPS. (✗)
4. Real-time constraints are more critical for a bike computer than for an engine control unit (ECU). (✗)
5. Engine Control Units (ECUs) primarily manage the entertainment system in a car. (✗)
6. Field-Programmable Gate Arrays (FPGAs) can be used to implement microprocessors within a System on a Chip (SoC). (✓)
7. A "soft core" processor is physically etched onto the silicon of an SoC. (✗)
8. The Internet of Things (IoT) involves distributed networks of embedded systems, often with sensors and controllers. (✓)

b) choose the best answer

1. What does IP stand for in the context of embedded systems?
a) Instruction Protocol | b) Internet Protocol
~~c)~~ Intellectual Property | d) Internal Peripheral

2. Which of the following is NOT a type of IP Core?
a) Hard IP Core | b) Soft IP Core
~~c)~~ Flexible IP Core | d) None of the above

3. A Soft IP Core is typically delivered as:
a) A physical chip | ~~b)~~ An encrypted or unreadable source file
c) A compiled binary file | d) A set of documentation

4. What is the role of EDA companies in the ARM ecosystem?
a) They design ARM processors. | ~~b)~~ They turn Soft IPs into Hard IPs.
c) They develop software for ARM processors. | d) They market ARM products.

5. The Cortex-M series is designed for which type of applications?
a) High-performance applications with complex OS | b) Real-time applications requiring high reliability
~~c)~~ Cost-sensitive microcontroller applications | d) Security-critical applications

6. Which Cortex series is suitable for applications like engine control units (ECUs)?
a) Cortex-A | ~~b)~~ Cortex-R
c) Cortex-M | d) SecurCore

7. Which of the following is an application of the Cortex-A series?
a) Smart sensors | ~~b)~~ Smartphones
c) Automotive braking systems | d) High-security access control

8. ARM7 processors are:
a) 64-bit | ~~b)~~ 32-bit
c) 16-bit | d) 8-bit

9. Which byte order stores the most significant byte first?
a) Little Endian | ~~b)~~ Big Endian
c) Mixed Endian | d) None of the above

10. Which of the following is an advantage of Little Endian?
a) Easier to print hexadecimal values | b) Easier to determine sign
~~c)~~ Easier for multi-precision arithmetic | d) Easier to compare numbers

12. What is the role of the SPSR?
a) Stores the status of the current program | ~~b)~~ Saves the CPSR value when entering exception modes
c) Points to the top of the stack | d) Stores the return address after a branch

14. Which mode is entered on an ARM7 processor immediately after a reset?
a) User Mode | ~~b)~~ Supervisor Mode
c) Interrupt Mode | d) Abort Mode

c) Match the following in pairs (one from column A with one from column B)

Column A	Column B	A→B
1. Hard IP Core	a. Scalable, delivered as source code (often encrypted)	2.......
2. Soft IP Core	b. High-performance processors for applications with OS support	.3......
3. Cortex-A Series	c. Pre-designed, physical block of logic; fixed electrical properties	1.........
4. Cortex-M Series	d. Cost-sensitive, deterministic microcontroller applications 4	....
5. ARM7TDMI's: the "D" sands for….	e. Hardware debugging	 5

Question 2: ARM assembly programming (8 marks)

a) Write ARM7 assembly program that finds the maximum value of int array of 100 items.

```
        AREA array_max, CODE, READONLY
        ENTRY
        ; Define array size and address
        ..LDR R1, =array.............// Load address of array into R1
        ....MOV R2, #100...........// Load array size (100) into R2
        ...LDR R0, [R1]...............// Initialize max (R0) with the first element of the array
loop    SUBS R2, R2, #1      ; Decrement counter, set flags
        ..BNE end.....................//    Branch to end if counter is zero
        ...LDR R3, [R1], #4..........//    Load the next element from the array into R3 (use post indexing)
        ..CMP R0, R3..................//    Compare current max (R0) with current element (R3)
        ...BGE next.....................//    If R0 >= R3, go to the next element in the array
        ....MOVLT R0, R3............//    If R3 > R0, update max with R3
next    B loop                       // Branch back to loop
end     // Maximum value is now in R0
stop    B stop             // Infinite loop to halt execution
        AREA data, DATA, READWRITE
          array  .DCD 1,2,3,4,5,6,7,8,9,10..//Example initial values (first 10)
        END
```

b) Create ARM7 Assembly 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]

```
        mov r0, #0x4      //r0=a0
        mov r1, #0x2      //r1=a1
        mov r2, #0x1      //r2=a2
        // Memory buffers
        x_buffer EQU 0x40000000  // start address of x[n]
        y_buffer EQU 0x40000100 //  start address of y[n]
        buffer_size EQU 97          // no of output samples to be computed
start:                  //Initialize pointers and counter
        ....LDR R3, =x_buffer.......//    r3 points to start address of x[n]
        ......LDR R4, =Y_buffer.....//    r4 points to start address of y[n]
        MOV     r5, #buffer_size  // r5 = Loop counter
loop:
        .....LDR R6, [R3].............// Get current input x[n] into r6 (replace with actual input reading)
        ......MUL R9, R0, R6........// y[n] = a0*x[n]       ; Calculate r9=y[n]
       // Access previous x samples (assuming 32-bit samples)
        LDR     r7, [r3, #-4]              // r7 = x[n-1]
        ..MLA R9, R9, R1, R7.......// y[n] += a1*x[n-1]
        ...LDR R8, [R3, #-4]..........// r8 = x[n-2]
        ....MLA R9, R9, R2, R8.....// y[n] += a2*x[n-2]
        STR     r9, [r4]                       // Store computed y[n]=r9 into its memory buffer
        //Increment pointers and decrement counter
        .ADD.....R3, R3, #4..........//  increment x[n] buffer pointer
        ......ADD....R4, R4, #4.....//  increment y[n] buffer pointer
        SUBS    r5, r5, #1   // decrement no of output samples to be computed
        BNE     loop
stop:   B      stop        // Infinite loop
```