USN | | | | | | | | | | |

Internal Assessment Test II – Aug 2023

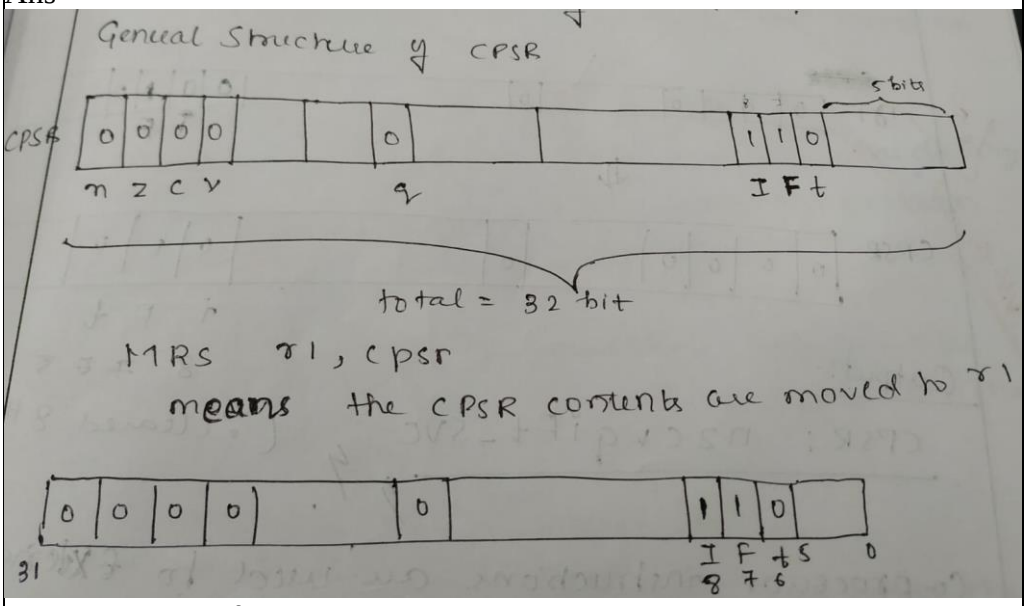
Sub:	MICROCONTROLLER AND EMBEDDED SYSTEMS					Sub Code:	21CS43	Branch:	CSE
Date:	11/8/2023	Duration:	90 mins	Max Marks:	50	Sem / Sec:	IV Sem A/B/C		OBE

Answer any FIVE FULL Questions	MARKS	CO	RBT
1. a)	[5+5]	CO2	L2,L3

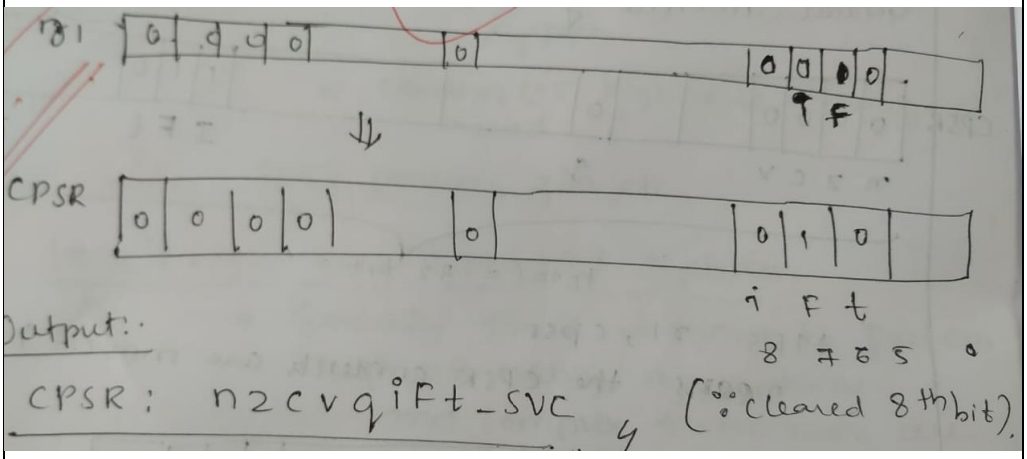
PRE-Condition
cpsr = nzcvqIFt_SVC

```
MRS  r1, cpsr
BIC  r1, r1, #0x80;
MSR  cpsr_c,r1
```

After execution of instruction what will be the status of cpsr?
Ans---



1.MRS r1, cpsr after executing this instruction CPSR content will move to r1
2. Next BIC r1, r1, #0x80 in this instruction, the binary equivalent 0x80 is 0b10000000, now BIC instruction will clear the 8th bit and stored in register r1.
3. MSR cpsr_c,r1 in this instruction the contents of r1 will move to CPSR.
So in CPSR interrupt flag will disable. So **cpsr = nzcvqiFt_SVC**



1.b)	Explain co-processor instructions with example.			
2.	What is the most efficient way to write a "For" Loop in ARMC compiler. Take one example, write C code and also compiler generated output, after that analyze the code.	[10]	CO2	L1,L2,L3

- Co-processors are used to increase / extend the ARM instruction set.
- Co-processors provide additional computational ability and other memory subsystem such as caches and memory management
- Co-processors provide 3 types of instructions:

1) Data processing instructions.
2) Register transfer instructions
3) Memory transfer instructions.

- The mnemonics are:

CDP : Data processing instruction
Performs operations in a coprocessor

MRC / MCR : Register transfer instruction
It transfers data to/ from coprocessor register

STC / LDC : Memory transfer instruction
It loads/ stores data into a coprocessor

- Syntax: CDP{cond} cp, opcode1, Cd, Cn{, opcode2}
  <MCR|MRC>{cond} cp, opcode1, Rd, Cn, Cm{, opcode2}
  <LDC|STC>{cond} cp, Cd, addressing
- Here cp - is the coprocessor number.
  Cd, Cn, Cm - are coprocessor registers
  Rd - destination register
  opcode1 - operation to be performed.
- Ex: ① MRC P15, 0, R1, C0, C0, 0
  cp, opcode1, Rd, Cn, Cm, opcode2.
  Contents of coprocessor registers is copied to general register
  ② CDP P15, 0, C0, C1, 0
  cp, opcode1, Cd, Cn, opcode2
  Required operation is performed on coprocessor.
  ③ LDC P15, C0, address
  cp, Cd, addressing
  Co-processor register is loaded

* The most efficient way of writing a "for" loop in ARMC compiler is by using a decrementing loop counter and using unsigned int or long datatype of loop counter. Also using "not equal to" instead of "greater than".
* The C-code for above type of "for" loop is given below:

```
int checksum_v6(int *data)
{
  unsigned int i;
  int sum=0;
  for(i=64; i!=0; i--)
  {
    sum += *(data++);
  }
  return sum;
}
```

* The corresponding compiler output:

```
checksum_v6
    MOV   r2, r0    ; r0 = data
    MOV   r0, #0    ; sum = 0
    MOV   r1, #0x40 ; i = 64.
checksum_v6_loop
    LDR   r3, [r2, r1, LSL #4] ; r3 = *(data++)
    SUBS  r1, r1, #1    ; i = i-1
    ADD   r0, r3, r0    ; sum += *(data++)
    BNE   checksum_v6_loop ; if i != 0
    MOV   pc, r14
```

→ only 4 instructions in the

* Analyzing the code:

→ Here since we use a decrementing loop, comp statement is not required. We can directly check status flag for zero flag.

→ Since we use unsigned int type of i, conversi into char type is not required. Therefore, the statement is reduced in the compiler output.

→ For an unsigned int loop counter, i!=0 and i>0 mean the same. But i>0 will generate the following code in compiler output:

```
SUBS  r1, r1, #1
CMP   r1, #0
BGT   LOOP
```

Hence the number of instructions increases.

→ Therefore, the above code checksum_v6 is the most optimised code using "for" loop.

3.(a)	What is Loop Unrolling? What will happen if the no of Loop iterations is not multiple of the Loop Unroll amount?	[6+4]	CO2	L1,L2,L3

a) * Loop unrolling is the concept of adding multiple instructions in the body of the loop specific number of times so that the number of loop iterations decreases.

* Each loop iteration requires 4 extra cycles for the execution of loop counter decrementation instruction and checking of branch instruction (conditional branch).
* In an ARM processor, one instruction is required to execute loop counter decrementation which requires one cycle of CPU.
* And branch requires 3 cycles of CPU.
* Therefore 4 cycles of CPU is needed for each iteration. This is called loop overhead.
* By unrolling the loop, we can reduced 4N cycles to (4N/N) = N cycles, if we unroll the loop 4 times.

(*) In number of loop iterations is a multiple of the unroll amount, we have the following example:

```
int checksum_v89(int *data, unsigned int N)
{
    int sum = 0;

    do {
        sum += *(data++);
        sum += *(data++);
        sum += *(data++);
        sum += *(data++);
        N -= 4;
    } while (N != 0);

    return sum;
}
```

Here, if N is a multiple of 4, then all the value in the data packet are added to sum.
But if N is not a multiple of 4, we have a solution below:

```
int checksum_v8(int *data, unsigned int N)
{ unsigned int i;
  int sum = 0;

  for (i = N/4; i != 0; i--)
  {
      sum += *(data++);
      sum += *(data++);
      sum += *(data++);
      sum += *(data++);
```

```
}
for(i=N&3; i!=0; i--)
{
    sum += *(data++);
}
return sum;
}
```

The checksum_v10 loop visits all the values in data packet, even the leftover cases are visited the second loop and added to sum.

(*) If the number of iterations is not a multiple of unroll amount then come up with a method to deal with the leftover cases as above.

What is pointer aliasing? Explain with example.

3) b) Pointer aliasing means two pointers pointing to the same address.

- A pointer acts as an alias to another pointer.
- In such cases, if one pointer is updated, the contents of the other is also affected.
- Hence it makes the code inefficient as the compiler has to load the pointer data every time an operation is to be performed.

Ex:

```
void time (int *time1, int *time2, int *state)
{
    *time1 += *state;
    *time2 += *state;
}
```

• Its equivalent compiler generated output would be

Here r0 → timer1 ; r1 → timer2 ; r2 → state

```
time_loop_function
    LDR  r4, [r0]    ;  r4 = *timer1
    LDR  r5, [r2]    ;  r5 = *state
    ADD  r4, r4, r5  ;  r4 = r4 + r5
    STR  r4, [r0]    ;  *timer1 = r4

    LDR  r4, [r1]    ;  r4 = *timer2
    LDR  r5, [r2]    ;  r5 = *state
    ADD  r4, r4, r5  ;  r4 = r4 + r5
    STR  r4, [r0]    ;  *timer2 = r4
```

• As we can see above, the load instruction to load state value is executed twice by the compiler although it is the same variable. This is because the compiler doesnot know if the pointers timer1 and timer2 are aliases. Hence the extra load instruction. Therefore, pointer aliasing makes the code inefficient and slow.

4.(a)	What is an embedded system? Differentiate between general purpose computing system and embedded system?	[2+4+4]	CO4	L1,L2

a) • An embedded system is a domain specific or application specific system developed to perform a specific task.
• Microcontrollers are used in embedded systems.
• Example for embedded systems are fridge, AC, microwave oven, etc.

b)

* General Purpose computing	* Embedded Systems
1) Developed and used for general computations purposes.	1) Developed and used for user and application specific domains
2) Need not be deterministic.	2) Needs to be deterministic.
3) Not at all / slightly tailored for performance increase.	3) Highly tailored for better performance.
4) Performance is the key factor in deciding the best system.	4) Domain specific performance like memory size, speed, etc used to decide the best system
5) Not designed / modified for low power consumption.	5) Designed specifically for low power consumption.
6) Used in general processing units	6) Used for a specific purpose
Ex: Computers processors, etc	Ex: AC, refrigerator, etc.

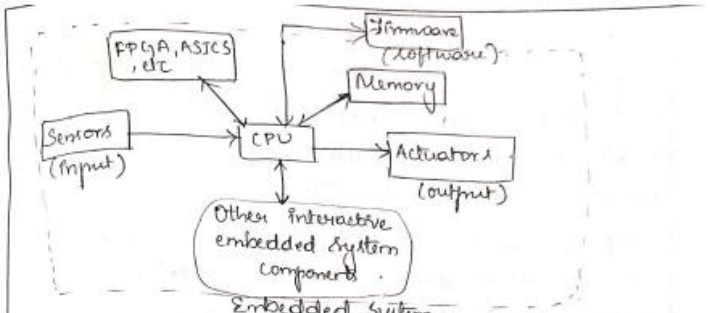
How can we classify the embedded system based on complexity and performance.

7) b) * Embedded systems can be classified into the following types based on complexity and performance

1) Small-scale embedded systems:
- Developed using 8/16-bit microprocessors or microcontrollers
- Has low cost
- Used for low-level processes
- It generally has no operating system for its functioning
- Ex: Electronic toy

	2) Medium-Scale embedded systems: - Developed using 16/32-bit microprocessor or microcontroller. - Has medium performance requirements. - It is complex than small-scale systems and also costlier. - It may have an operating system for its functioning. 3) Large-Scale Embedded systems: - Developed using 32/64-bit microprocessor or microcontroller or DSP. - Has very complex structure. - Very costly. - Satisfies high-performance requirements. - It has an operating system (Real-time operating system) - RTOS.			
5.	List any four purposes of embedded system with examples. Ans-Embedded systems are used in various domains like consumer electronics, home automation, telecommunications, automotive industry, healthcare, control & instrumentation, retail and banking applications, etc. ***Data Collection, Storage, Representation*** ❑ Embedded systems with analog data capturing techniques collect data directly in the form of analog signal; whereas embedded systems with digital data collection mechanism convert the analog signal to corresponding digital signal using analog to digital (A/D) converters. ❑ If the data is digital, it can be directly captured by digital embedded system. ■ A digital camera is a typical example of an embedded system with data collection, storage, and representation of data. Images are captured and captured image may be stored within the memory of the camera. The captured image can also be presented to the user through a graphic LCD (Liquid Crystal Display) unit. ***Data (Signal) Processing*** ❑ Embedded systems with signal processing functionalities are employed in applications demanding signal processing like speech coding, audio-video codec, transmission applications, etc. ■ A digital hearing aid is a typical example of an embedded system employing data processing. ***Monitoring*** ❑ Almost all embedded products coming under the medical domain are with monitoring functions.	[10]	CO4	L1,L2

	■ Patient heart beat is monitored by Electro cardiogram (ECG) machine. Digital CRO, digital multi-meters, and logic analysers are examples of monitoring embedded systems ***Application Specific User Interface*** ❑ These are embedded systems with application-specific user interfaces like buttons, switches, keypad, lights, bells, display units, etc. ❑ Mobile phone is an example for this. ■ In mobile phone, the user interface is provided through the keypad, graphic LCD module, system speaker, vibration alert, etc.			
6 .	Which are the components used as the core of an embedded system? Explain the different units of Digital Signal Processor. 1). * The components used at the core of an embedded system are : 1) General purpose and domain specific processors. - microprocessor. - microcontrollers. - digital signal processors (DSPs). 2) Programmable Logic Devices (PLDs). 3) Application Specific Integrated Circuit (ASIC).	[6+4]	CO4	L1,L2



Embedded System.

* Core of the embedded system.

* Digital Signal Processors (DSPs):

→ DSP is the most efficient processor among all general purpose and domain specific processors. (microprocessor and microcontroller).

→ It is a domain-specific processor which is 2 or 3 times more faster than microprocessors and microcontrollers.

→ It implements the program on the hardware and hence faster, as the general purpose processors implement it on software.

→ The DSP is divided into 4 main units:

1) Program Memory: The memory files containing the program data on how to process the input signals is called program memory.

2) Data Memory: The input signals from the environment which may be analog or digital is called data memory. It is sensed by the sensors and processed by the program memory.

3) Computational unit: All the arithmetic and logical computations such as "addition", "subtraction", "multiplication" and "division" are performed here.

4) I/O unit: The interface of the DSP with the outside world is called the I/O unit.

→ DSP is capable of performing complex mathematical computations such as fast fourier transform (FFT), Discrete fourier transform (DFT), sum of products (SOP), etc.

→ DSP is a very fast and efficient application/domain specific processor.

Faculty Signature CCI Signature HOD Signature