



USN <input type="text"/>									
Internal Assessment Test 2– August 2022									
Sub:	Microcontroller and Embedded Systems				Sub Code:	18CS44	Branch	ISE	
Date:	08/08/2022	Duration:	90 min's	Max Marks:	50	Sem/Sec:	IV / A, B and C		OBE
Answer any FIVE FULL Questions							MARKS	CO	RBT
1	Explain the classification of embedded system.					10	CO3	L2	
2	Mention the applications of embedded system with example of each					10	CO3	L2	
3	Define the term RAM. Mention the different types of RAM. Explain anyone with neat circuit diagram.					10	CO3	L2	
4	a	What are programmable logic devices? Compare CPLD and FPGA.				07	CO3	L2	
	b	List the advantages of PLD over fixed logic devices.				03	CO3	L2	
5	With a neat interfacing diagram and waveform, explain I2C bus					10	CO3	L2	
6	Write a C program that prints the squares of integers between 0 to 9 using function and explain how to convert this C function to an assembly function					10	CO2	L3	

Faculty Signature

CCI Signature

HOD Signature

USN <input type="text"/>									
Internal Assessment Test 2– August 2022									
Sub:	Microcontroller and Embedded Systems				Sub Code:	18CS44	Branch	ISE	
Date:	08/08/2022	Duration:	90 min's	Max Marks:	50	Sem/Sec:	IV / A, B and C		OBE
Answer any FIVE FULL Questions							MARKS	CO	RBT
1	Explain the classification of embedded system.					10	CO3	L2	
2	Mention the applications of embedded system with example of each					10	CO3	L2	
3	Define the term RAM. Mention the different types of RAM. Explain anyone with neat circuit diagram.					10	CO3	L2	
4	a	What are programmable logic devices? Compare CPLD and FPGA.				07	CO3	L2	
	b	List the advantages of PLD over fixed logic devices.				03	CO3	L2	
5	With a neat interfacing diagram and waveform, explain I2C bus					10	CO3	L2	
6	Write a C program that prints the squares of integers between 0 to 9 using function and explain how to convert this C function to an assembly function					10	CO2	L3	

Faculty Signature

CCI Signature

HOD Signature

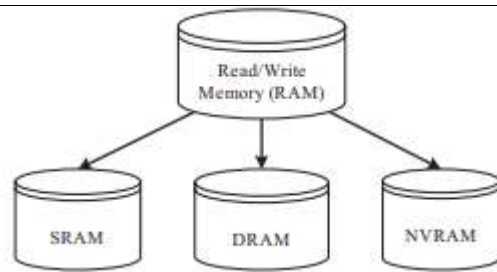
Internal Assessment Test 1 – July 2022

Scheme of Evaluation

Sub:	Microcontroller and Embedded Systems				Sub Code:	18CS44	Branch:	ISE		
Date:	07/07/2022	Duration:	90 min's	Max Marks:	50	Sem/Sec:	VI / A, B & C	OBE		
<u>Answer any FIVE FULL Questions</u>								MARKS	CO	RBT
1	<p>Explain classification of embedded system.</p> <p>Scheme: - Types + Explanation = 3+7 M = 10M</p> <p>Solution:-</p> <p>It is possible to have a multitude of classifications for embedded systems, based on different criteria. Some of the criteria used in the classification of embedded systems are as follows:</p> <ol style="list-style-type: none"> (1) Based on generation (2) Complexity and performance requirements (3) Based on deterministic behavior (4) Based on triggering. <p>The classification based on deterministic system behavior is applicable for 'Real Time' systems. The application/task execution behavior for an embedded system can be either deterministic or non-deterministic. Based on the execution behavior, Real Time embedded systems are classified into Hard and Soft. Embedded Systems which are 'Reactive' in nature (Like process control systems in industrial control applications) can be classified based on the trigger. Reactive systems can be either event triggered, or time triggered.</p> <p>(3)</p> <p>Explanation of all classification</p> <p>1.Classification Based on Generation:</p> <p>This classification is based on the order in which the embedded processing systems evolved from the first version to where they are today. As per this criterion, embedded systems can be classified into the following:</p> <p>1.1First Generation:</p> <p>The early embedded systems were built around 8bit microprocessors like 8085 and Z80, and 4bit microcontrollers. Simple in hardware circuits with firmware developed in Assembly code. Digital telephone keypads, stepper motor control units etc. are examples of this.</p> <p>1.2 Second Generation</p> <p>These are embedded systems built around 16bit microprocessors and 8 or 16 bit microcontrollers, following the first generation embedded systems. The instruction set for the second generation processors/controllers were much more complex and powerful than the first generation processors/ controllers. Some of the second generation embedded systems contained embedded operating systems for their operation. Data Acquisition Systems, SCADA systems, etc. are examples of second generation embedded systems.</p> <p>1.3 Third Generation</p> <p>With advances in processor technology, embedded system developers started making use of powerful 32bit processors and 16bit microcontrollers for their design. A new concept of application and domain specific processors/controllers like Digital Signal Processors (DSP) and Application Specific Integrated Circuits (ASICs) came into</p>						[10]	CO1	L2	

	<p>the picture. The instruction set of processors became more complex and powerful and the concept of instruction pipelining also evolved. The processor market was flooded with different types of processors from different vendors. Processors like Intel Pentium, Motorola 68K, etc. gained attention in high performance embedded requirements. Dedicated embedded real time and general-purpose operating systems entered into the embedded market. Embedded systems spread its ground to areas like robotics, media, industrial process control, networking, etc.</p> <p>1.4 Fourth Generation</p> <p>The advent of System on Chips (SoC), reconfigurable processors and multicore processors are bringing high performance, tight integration, and miniaturisation into the embedded device market. The SoC technique implements a total system on a chip by integrating different functionalities with a processor core on an integrated circuit. We will discuss about SoCs in a later chapter. The fourth-generation embedded systems are making use of high-performance real time embedded operating systems for their functioning. Smart phone devices, mobile internet devices (MIDs), etc. are examples of fourth generation embedded systems.</p> <p>2.Classification Based on Complexity and Performance</p> <p>This classification is based on the complexity and system performance requirements. According to this classification, embedded systems can be grouped into the following</p> <p>2.1 Small-Scale Embedded Systems</p> <p>Embedded systems which are simple in application needs and where the performance requirements are not time critical fall under this category. An electronic toy is a typical example of a small-scale embedded system. Small-scale embedded systems are usually built around low performance and low cost 8- or 16-bit microprocessors/microcontrollers. A small-scale embedded system may or may not contain an operating system for its functioning.</p> <p>2.2 Medium-Scale Embedded Systems</p> <p>Embedded systems which are slightly complex in hardware and firmware (software) requirements fall under this category. Medium-scale embedded systems are usually built around medium performance, low cost 16- or 32-bit microprocessors/microcontrollers or digital signal processors. They usually contain an embedded operating system (either general purpose or real time operating system) for functioning.</p> <p>2.3 Large-Scale Embedded Systems/Complex Systems</p> <p>Embedded systems which involve highly complex hardware and firmware requirements fall under this category. They are employed in mission critical applications demanding high performance. Such systems are commonly built around high performance 32- or 64-bit RISC processors/controllers or Reconfigurable System on Chip (RSoC) or multi-core processors and programmable logic devices. They may contain multiple processors/controllers and co-units/hardware accelerators for offloading the processing requirements from the main processor of the system. Decoding/ encoding of media, cryptographic function implementation, etc. are examples for processing requirements which can be implemented using a co-processor/hardware accelerator. Complex embedded systems usually contain a high-performance Real-Time Operating System (RTOS) for task scheduling, prioritization, and management.</p> <p style="text-align: right;">(7)</p>			
2	<p>Mention the applications of embedded system with example of each Scheme: - Applications – 10 Marks</p> <p>Solution:-</p>	[10]	CO1	L2

	<p>Embedded technology has acquired a new dimension from its first generation model, the Apollo guidance computer, to the latest radio navigation system combined with in-car entertainment technology and the wearable computing devices (Apple watch, Microsoft Band, Fitbit fitness trackers etc.). The application areas and the products in the embedded domain are countless. A few of the important domains and products are listed below:</p> <ol style="list-style-type: none"> (1) Consumer electronics: Camcorders, cameras, etc. (2) Household appliances: Television, DVD players, washing machine, fridge, microwave oven, etc. (3) Home automation and security systems: Air conditioners, sprinklers, intruder detection alarms, closed circuit television cameras, fire alarms, etc. (4) Automotive industry: Anti-lock braking systems (ABS), engine control, ignition systems, automatic navigation systems, etc. (5) Telecom: Cellular telephones, telephone switches, handset multimedia applications, etc. (6) Computer peripherals: Printers, scanners, fax machines, etc. (7) Computer networking systems: Network routers, switches, hubs, firewalls, etc. (8) Healthcare: Different kinds of scanners, EEG, ECG machines etc. (9) Measurement & Instrumentation: Digital multimeters, digital CROs, logic analyzers PLC systems, etc. (10) Banking & Retail: Automatic teller machines (ATM) and currency counters, point of sales (POS) (11) Card Readers: Barcode, smart card readers, handheld devices, etc. (12) Wearable Devices: Health and Fitness Trackers, Smartphone Screen extension for notifications, etc. (13) Cloud Computing and Internet of Things (IOT) (10) 			
3	<p>Define the term RAM. Mention the different types of RAM. Explain anyone with neat circuit diagram.</p> <p>Scheme: - (i) Define the term RAM & Mention the different types of RAM – 5 Marks</p> <p style="padding-left: 40px;">(ii) . Explain anyone with neat circuit diagram. - 5 Marks</p> <p>Solution: - (i) 5 Marks</p> <p>RAM is the data memory or working memory of the controller/processor. Controller/processor can read from it and write to it. RAM is volatile, meaning when the power is turned off, all the contents are destroyed. RAM is a direct access memory, meaning we can access the desired memory location directly without the need for traversing through the entire memory locations to reach the desired memory position (i.e. random access of memory location). This is in contrast to the Sequential Access Memory (SAM), where the desired memory location is accessed by either traversing through the entire memory or through a ‘seek’ method. Magnetic tapes, CD ROMs, etc. are examples of sequential access memories. RAM generally falls into three categories: Static RAM (SRAM), dynamic RAM (DRAM), and non-volatile RAM (NVRAM).</p>	[10]	CO1	L2



Classification of Working Memory (RAM)

Solution: - (ii)

Explanation of any one RAM with neat diagram: 5 Marks.

4(a) What are programmable logic devices? Compare CPLD and FPGA [10] CO1 L2

Scheme: (i) Define PLD + Compare CPLD and FPGA – 7 Marks

Solution: -

(i) Logic devices provide specific functions, including device-to-device interfacing, data communication, signal processing, data display, timing and control operations, and almost every other function a system must perform. **(2)**

(ii) The two major types of programmable logic devices are Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs). Of the two, FPGAs offer the highest amount of logic density, the most features, and the highest performance. The largest FPGA now shipping, part of the Xilinx Virtex™§ line of devices, provides eight million “system gates” (the relative density of logic). These advanced devices also offer features such as built-in hardwired processors (such as the IBM power PC), substantial amounts of memory, clock management systems, and support for many of the latest, very fast device-to-device signaling technologies. FPGAs are used in a wide variety of applications ranging from data processing and storage to instrumentation, telecommunications, and digital signal processing. CPLDs, by contrast, offer much smaller amounts of logic—up to about 10,000 gates. But CPLDs offer very predictable timing characteristics and are therefore ideal for critical control applications. CPLDs such as the Xilinx CoolRunner™† series also require extremely low amounts of power and are very inexpensive, making them ideal for cost-sensitive, battery-operated, portable applications such as mobile phones and digital handheld assistants.

(7)

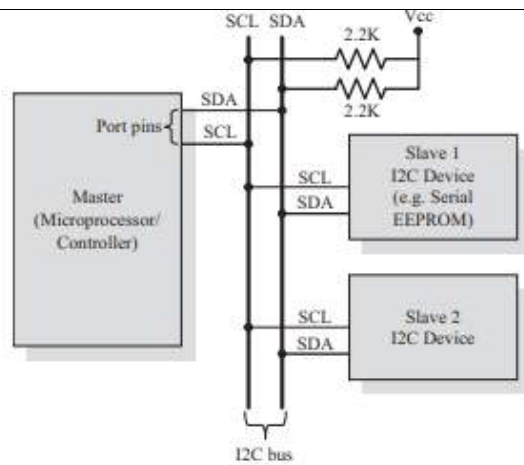
4(b) List the advantages of PLD over fixed logic devices.

Scheme: - 3 Marks

Solution:

Advantages of PLD Programmable logic devices offer a number of important advantages over fixed logic devices, including the following: (1) PLDs offer customers much more flexibility during the design cycle because design iterations are simply a matter of changing the programming file, and the results of design changes can be seen immediately in working parts. (2) PLDs do not require long lead

	<p>times for prototypes or production parts—the PLDs are already on a distributor’s shelf and ready for shipment. (3) PLDs do not require customers to pay for large NRE costs and purchase expensive mask sets—PLD suppliers incur those costs when they design their programmable devices and are able to amortize those costs over the multi-year lifespan of a given line of PLDs. (4) PLDs allow customers to order just the number of parts they need, when they need them, allowing them to control inventory. Customers who use fixed logic devices often end up with excess inventory which must be scrapped, or if demand for their product surges, they may be caught short of parts and face production delays. (5) PLDs can be reprogrammed even after a piece of equipment is shipped to a customer. In fact, thanks to programmable logic devices, a number of equipment manufacturers now tout the ability to add new features or upgrade products that already are in the field. To do this, they simply upload a new programming file to the PLD, via the Internet, creating new hardware logic in the system. (3)</p>			
5	<p>With a neat interfacing diagram and waveform, explain I2C bus Scheme:- I2C bus definition + Explanation = 10 Marks Solution:- The Inter Integrated Circuit Bus (I2C—Pronounced ‘I square C’) is a synchronous bi-directional half duplex (one-directional communication at a given point of time) two wire serial interface bus. The concept of I2C bus was developed by ‘Philips semiconductors’ in the early 1980s. The original intention of I2C was to provide an easy way of connection between a microprocessor/ microcontroller system and the peripheral chips in television sets. The I2C bus comprise of two bus lines, namely, Serial Clock—SCL and Serial Data—SDA. SCL line is responsible for generating synchronization clock pulses and SDA is responsible for transmitting the serial data across devices. I2C bus is a shared bus system to which many number of I2C devices can be connected. Devices connected to the I2C bus can act as either ‘Master’ device or ‘Slave’ device. The ‘Master’ device is responsible for controlling the communication by initiating/terminating data transfer, sending data and generating necessary synchronization clock pulses. ‘Slave’ devices wait for the commands from the master and respond upon receiving the commands. ‘Master’ and ‘Slave’ devices can act as either transmitter or receiver. Regardless of whether a master is acting as transmitter or receiver, the synchronization clock signal is generated by the ‘Master’ device only. I2C supports multi masters on the same bus. The following bus interface diagram shown in Fig. illustrates the connection of master and slave devices on the I2C bus.</p>	[06]	CO2	L2



12C Bus Interfacing

The I2C bus interface is built around an input buffer and an open drain or collector transistor. When the bus is in the idle state, the open drain/collector transistor will be in the floating state and the output lines (SDA and SCL) switch to the 'High Impedance' state. For proper operation of the bus, both the bus lines should be pulled to the supply voltage (+5V for TTL family and +3.3V for CMOS family devices) using pull-up resistors. The typical value of resistors used in pull-up is 2.2K. With pull-up resistors, the output lines of the bus in the idle state will be 'HIGH'. The address of a I2C device is assigned by hardwiring the address lines of the device to the desired logic level. The address to various I2C devices in an embedded device is assigned and hardwired at the time of designing the embedded hardware. The sequence of operations for communicating with an I2C slave device is listed below:

1. The master device pulls the clock line (SCL) of the bus to 'HIGH'
2. The master device pulls the data line (SDA) 'LOW', when the SCL line is at logic 'HIGH' (This is the 'Start' condition for data transfer)
3. The master device sends the address (7 bit or 10 bit wide) of the 'slave' device to which it wants to communicate, over the SDA line. Clock pulses are generated at the SCL line for synchronizing the bit reception by the slave device. The MSB of the data is always transmitted first. The data in the bus is valid during the 'HIGH' period of the clock signal
4. The master device sends the Read or Write bit (Bit value = 1 Read operation; Bit value = 0 Write operation) according to the requirement
5. The master device waits for the acknowledgement bit from the slave device whose address is sent on the bus along with the Read/Write operation command. Slave devices connected to the bus compares the address received with the address assigned to them.

	<p>6. The slave device with the address requested by the master device responds by sending an acknowledge bit (Bit value = 1) over the SDA line</p> <p>7. Upon receiving the acknowledge bit, the Master device sends the 8bit data to the slave device over SDA line, if the requested operation is 'Write to device'. If the requested operation is 'Read from device', the slave device sends data to the master over the SDA line</p> <p>8. The master device waits for the acknowledgement bit from the device upon byte transfer complete for a write operation and sends an acknowledge bit to the Slave device for a read operation</p> <p>9. The master device terminates the transfer by pulling the SDA line 'HIGH' when the clock line SCL is at logic 'HIGH' (Indicating the 'STOP' condition) The first generation I2C devices were designed to support data rates only up to 100kbps. Over time there have been several additions to the specification so that there are now five operating speed categories; Namely, Standard mode (Sm - Data rate up to 100kbit/sec), Fast mode (Fm - Data rate up to 400kbit/sec), Fast mode Plus (Fm+ - Data rate up to 1Mbit/sec), and High-speed mode (Hs-mode - Data rate up to 3.4Mbit/sec) and an Ultra Fast-mode (UFm), with a bit rate up to 5 Mbit/s for unidirectional I2C bus. (10)</p>			
6	<p>Write a C program that prints the squares of integers between 0 to 9 using function and explain how to convert this C function to an assembly function</p> <p>Scheme: - Program – 10 marks</p> <p>Solution: -</p> <pre> #include <stdio.h> int square(int i); int main(void) { int i; for (i=0; i<10; i++) { printf("Square of %d is %d\n", i, square(i)); } } int square(int i) </pre>	[06]	CO1	L1


```
{  
    return i*i;  
}
```

To replace square by an assembly function that performs the same action. Remove the C definition of square, but not the declaration (the second line) to produce a new C file main1.c. Next add an armasm assembler file square.s with the following contents.

```
        AREA    |.text|, CODE, READONLY  
  
        EXPORT  square  
  
        ; int square(int i)  
square  
    MUL    r1, r0, r0    ; r1 = r0 * r0  
    MOV    r0, r1        ; r0 = r1  
    MOV    pc, lr        ; return r0  
    END
```

The AREA directive names the area or code section that the code lives in. If you use nonalphanumeric characters in a symbol or area name, then enclose the name in vertical bars. Many nonalphanumeric characters have special meanings otherwise. In the previous code we define a read-only code area called .text. The EXPORT directive makes the symbol square available for external linking. At line six we define the symbol square as a code label. Note that arm. ARM treats no indented text as a label definition. When square is called, the parameter passing is defined by the ATPCS. The input argument is passed in register r0, and the return value is returned in register r0. The multiply instruction has a restriction that the destination register must not be the same as the first argument register. Therefore, we place the multiply result into r1 and move this to r0. The END directive marks the end of the assembly file. **(10)**

Faculty Signature

CCI Signature

HOD Signature