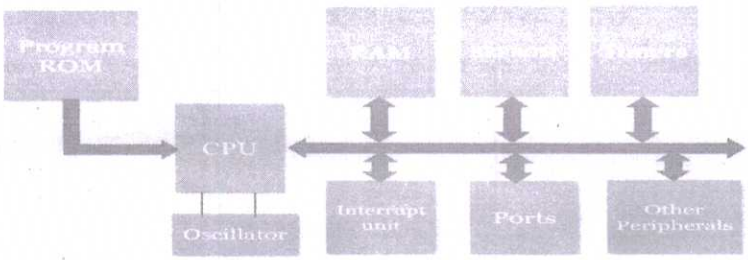
REVISION : 2015 COURSE CODE : 5041

COURSE TITLE: EMBEDDED SYSTEMS

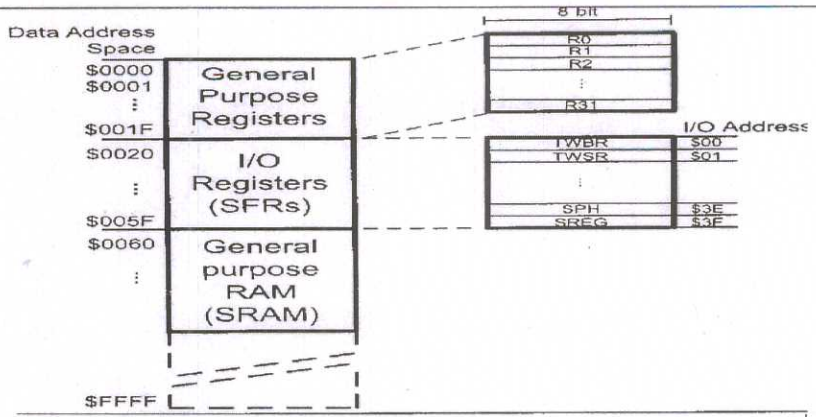
Question no	Scoring indicators	Split up score	Sub total	Total
	PART – A			
I 1.	RAM-2K ROM-32K		2	
2.	Assembler Directives are not instruction, but instructions to the assembler before a program is assembled.		2	10
3.	Serializing data is a way of sending a byte of data one bit at a time through a single pin of microcontroller.		2	
4.	The process of saving the context of a task being suspended and restoring the context of a task being resumed is called context switching		2	
5	The Kernal is a computer program that constitutes the central core of a computer's operating system. It is the first program that is loaded on start up		2	
II . 1	PART – B • 32 x 8 general working purpose registers. • 32K bytes of in system self programmable flash program memory. • 2K bytes of internal SRAM. • 1024 bytes EEPROM. • Available in 40 pin DIP, 44 lead QTFP, 44-pad QFN/MLF. • 32 programmable I/O lines. • 8 Channel, 10 bit ADC		6	

2.	• AVR as an EEPROM memory that is used for storing data. • EEPROM does not lose it's data when power id off where as SRAM lose its data when power is on • EEPROM is used for storing data that should rarely be changed whereas the SRAM is used for storing data and parameters that are changed frequently. • In AVR data sheet EEPROM refers to EEPROM's size and SRAM is the internal SRAM's size.		6	
3	• Hexadecimal numbers –Put 0X in front of the number or put $ symbol in front of the number • Binary numbers- eg: LDI R16, 0b10001101 • Decimal numbers- eg: LDI R23,12 • ASCII characters- eg: LDI R20, '9'		6	
4.	• Macros increases code size every time they are invoked • For eg if you call a 10-instruction macro 10 times the code size is increased by 100 instruction where as if you call a subroutine 10 times the code size is only that of the subroutine instruction • On the other hand a function call takes 3 or 4 clo ck and the instruction takes 4 clock to get executed • The subroutine use stack space as when called while the macros do not		6	30

5	#include <avr/io.h>
	int main (void)
	{
	unsigned char x,y;
	unsigned char mybyte=0x29;
	DDRB=DDRC=0xFF;
	x=mybyte&0x0F;
	PORTB=x|0x30;
	y=mybyte&0xF0;
	y=y>>4;
	PORTC=y|0x30;
	return 0;
	}
			6	

6.	• If two interrupts are activated at the same time, the interrupt with the higher priority is served first. • The priority of each interrupt is related to the address of that interrupt in the interrupt vector . • The interrupt that has a lower address, has a higher priority . • For example, the address of external interrupt 0 is 2 while, the address of external interrupt 2 is 6; thus, external interrupt 0 has a higher priority, and if both of these interrupts are activated at the same time , external interrupt 0 is served first		6	
7.	• Embedded systems do a very specific task, they cannot be programmed to do different things. • Have very limited resources • They have to work against some deadlines • They are constrained for power. Power consumption has to be very low • They need to be highly reliable • Have to operate in extreme climatic conditions		6	
III. (a)	PART -- C BASIC ARCHITECTURE  Explanation	5 4	9	

III. (b)



3

6 | 15

- The I/O memory is dedicated to specific function such as status register, timers,serial communication, I/O ports. ADC, and so on
- The function of each I/O memory location is fixed by the CPU designer at the time of design because it is used for control of the microcontroller or peripherals

3

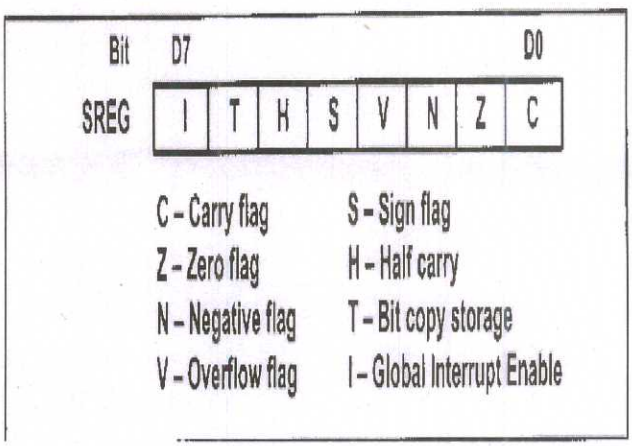
IV. (a)

- These various ways of accessing data are called *addressing modes*.
- 1. Single – Register (Immediate)
- 2. Register.
- 3. Direct.
- 4. Register indirect.
- 5. Flash direct
- 6. Flash indirect

3

1*6 | 9

IV (b)



15

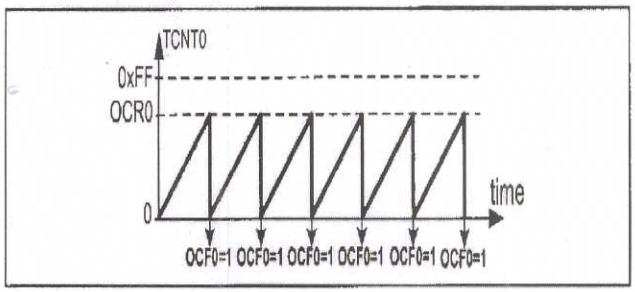
6

V. (a)	• First we use a text editor to type the program • We use AVR Studio IDE which has a text editor assembler simulator and much more all in one software • The editor must be able to produce ASCII file • For assemblers the file follow the DOS convention but the source file has the extension "asm" • The "asm" source file containing the program code created in step 1 is fed to the AVR assembler		8	15
V (b)	ADD, ADC,SUB,SBC,SUBI,SBCI,SBIW,MUL (Any seven)	1*7	7	
VI (a)	Call instruction is a control transfer instruction which is used to call a subroutine . • In AVR there are 4 instruction to call subroutine . • CALL (long call), RCALL (relative call), ICALL (indirect call to Z), EICALL (extended indirect call to Z).		8	
VI (b)	LDI R16,200 SBI DDRB,2 AGAIN: SBI PORTB,2 CBI PORTB,2 DEC R16 BRNE AGAIN		7	15
VII (a)	Bitwise operators :AND&, OR|, EX-OR^, INVERTER~ Compound Assignment Operators: &=, |=	(2*4)	8	

(b)	#include <avr/io.h>
	int main(void)
	{
	unsigned char x,binbyte,d1,d2,d3;
	DDRB=DDRC=DDRD=OxFF;
	binbyte=0xFD;
	x=binbyte/10;
	d1=binbyte%10;
	d2=x%10;
	d3=x/10;
	PORTB=d1;
	PORTC=d2;
	PORTD=d3;
	Return 0;
	}
			7	15

VIII (a)

- The OCR0 register is used with CTC mode .
- As with the normal mode, in the CTC mode, the timer is incremented with a clock.
- But it counts up until the content of the TCNT0 register becomes equal to the content of OCR0 (compare match occurs); then, the timer will be declared and the OCF0 flag will be set when the next clock occurs. 5
- OCF0 flag is located in the TIFR register.



4

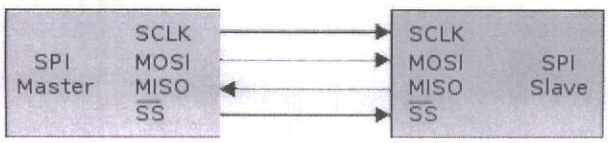
9

VIII (b)

- Inter –integrated circuit bus uses 2 wires for connecting devices. The bus is bidirectional and synchronous to a common clock.
- The I2C bus uses two wires: serial data (SDA) and serial clock (SCL). All I2C master and slave devices are connected with only those two wires. Each device can be a transmitter, a receiver or both. Some devices are masters – they generate bus clock and initiate communication on the bus, other devices are slaves and respond to the commands on the bus. In order to communicate with specific device, each slave device must have an address which is unique on the bus. I2C master devices (usually microcontrollers) don't need an address since no other (slave) device sends commands to the master

6

15



Single Main to single Secondary: basic SPI bus example

The **Serial Peripheral Interface** (**SPI**) is a synchronous serial communication interface specification used for short-distance communication, primarily in embedded systems. The interface was developed by Motorola in the mid-1980s and has become a *de*

	facto standard. Typical applications include Secure Digital cards and liquid crystal displays. SPI devices communicate in full duplex mode using a master-slave architecture (alternate terminology being main and secondary) with a single master. The master device originates the frame for reading and writing. Multiple slave-devices are supported through selection with individual slave select (SS), sometimes called chip select (CS), lines. Sometimes SPI is called a *four-wire* serial bus, contrasting with three-, two-, and one-wire serial buses. The SPI may be accurately described as a synchronous serial interface, but it is different from the Synchronous Serial Interface (SSI) protocol, which is also a four-wire synchronous serial communication protocol. The SSI protocol employs differential signaling and provides only a single simplex communication channel. SPI is one master and multi slave communication.			
IX (a)	1. Consumer Appliances 2. Office Automation 3. Industrial automation 4. Medical Electronics 5. Telecommunications 6. Wireless Technologies 7. Security 8. Finance		9	
IX (b)	***Single System Control Loop*** Single system control loop is the simplest type of embedded operating system. It is so like operating system but it is designed to run the only single task. It still under debate that this system should be classified as a type of operating system or not. ***Multi-Tasking Operating System*** As the name suggests that this operating system can perform multiple tasks. In multi-tasking operating system there are several tasks and processes that execute simultaneously. More than one function can be performed if the system has more than one core or processor. The operating system is switched between tasks. Some tasks wait for events while other receive events and become ready to run. If one is using a multitasking operating system, then software development is simplified because different components of software can be made independent to each other. ***Rate Monotonic Operating System*** It is a type of operating system that ensures that task runs in a			

		system can run for a specific interval of time and for a specific period of time. When it is not ensured, there comes a notification of failure to system software to take suitable action. This time limit cannot be ensured if the system is oversubscribed, at this point another event may occur during run time and the failure notification comes. ***Preemptive Operating System*** A preemptive operating system is a type of multitasking operating system that interprets the preemptive predominance for tasks. A higher priority is task is always defined and run before a lower priority task. Such multi-tasking operating systems are efficient in increasing system response to events and also simplify the development of software making the system more reliable. The designer of the system may be able to calculate the time required for the service interprets in a system and also the time is taken by the scheduler for switching tasks. Such systems may fail to meet the deadline of a system and the software is unaware of the missed deadline. CPU loading in a preemptive operating system can be measured naturally by defining a lower priority task that only increments counter and do nothing else. ***Real Time Operating System*** A real-time operating system is the one which serves real time applications. It processes data as it comes in. The time requirements for processing of operating system are usually measured in shorter increments or in 10^{th} of seconds. They may be time sharing or driven by events. Real time Operating systems are used in small embedded systems. The main features of real-time operating system include • Process threads that can be defined • Multitasking • Interrupt levels		6	15
X	(a)	Raspberry Pi is a credit card sized single computer board. It can be used for many of the things that your desktop PC does, like spreadsheets, word-processing, games and it can also play high definition video. It was developed byRaspberry Pi foundation from UK. ... The raspberry picomputer is portable and less expensive.		8	15
X	(b)	■ Operating System can be categorized into ■ Non real time OS: These operating systems have a small footprint but they are not suitable for hard real time application . ■ These OS are used in consumer appliances such as DVD, set top box etc. are soft real time embedded systems.		7	