Scoring Indicators

Course : Embedded System **Code : TED (15) 5041**

Revision : 2015

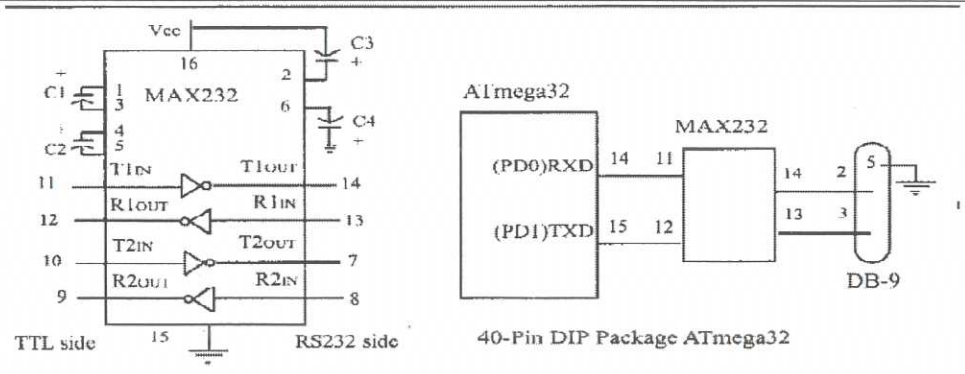
Question No.		Scoring Indicators	Split Score	Total Score
I.	1.	▪ 4K to 256K program memory ▪ 28 to 100 pins package ▪ Rich instruction set (more than 120 instructions) ▪ Extensive peripheral set.	1+1	2
I.	2	• AND Rd, Rr ;Rd=Rd AND Rr • Performs logical AND and place the result in left hand operand • ANDI Rd,K • Affect Z,S and N flag • N is D7 of the result and Z=1 if the result is zero • OR Rd, Rr ;Rd=Rd OR Rr • Performs logical OR and place the result in left hand operand • ORI Rd,K • Affect Z,S and N flag • N is D7 of the result and Z=1 if the result is zero	1+1 (Any two)	2
I.	3	*ROM -32K,RAM-2K*	1+1	2
I.	4	**Table 7-4: Bit-wise Shift Operators for C** Operation | Symbol | Format of Shift Operation Shift right | >> | data >> number of bits to be shifted right Shift left | << | data << number of bits to be shifted left)	1+!	2
I.	5	■ Tasks may need to exchange data amongst themselves. For instance, a task may write some data to a file and another task has to read the data. ■ This mechanism is known as inter-task communication	2	2
II.	1	• 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	6

<table>
<tr><td>II.</td><td>2</td><td>

Call instruction is a control transfer instruction which is used to call a subroutine .
In this 4 byte instruction , 10 bits are used for the op codes and the other 22 bit ,K21 to K0 are used for the address for the target subroutine.
Therefore ,it can be used to call subroutine locates any where within the 4M address space of 000000-$3FFFFF.
After the execution of the subroutine AVR should execute the next instruction .
For that , AVR saves the address of the instruction immediately below the call on the stack.
When a sub routine is called the control is transferred to that subroutine and the processor saves the PC of the next instruction on the stack and begins to fetch instruction from the new location.
After finishing the subroutine the RET instruction transfers control back to the caller.

</td><td>6</td><td>6</td></tr>
<tr><td>II.</td><td>3</td><td>

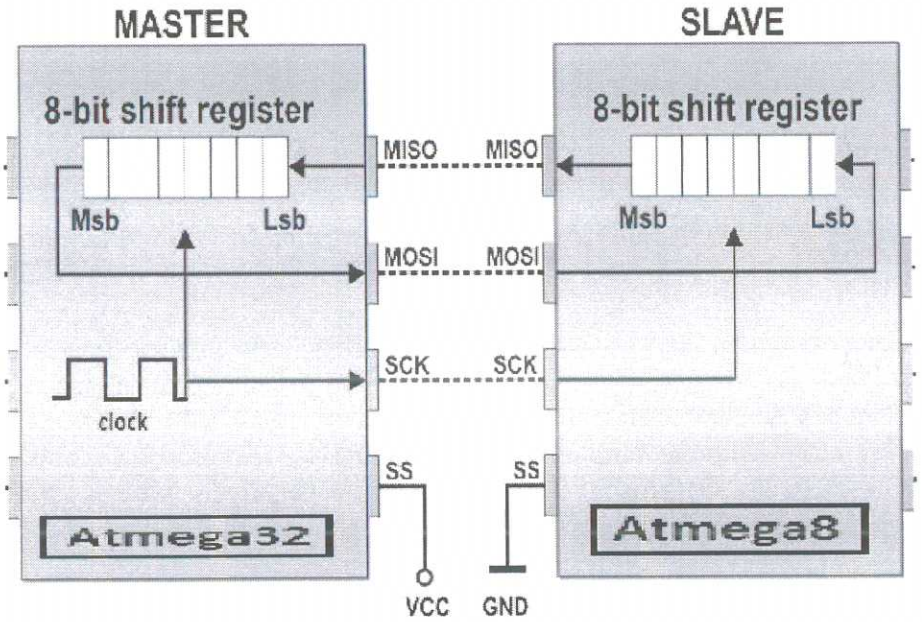
```
.INCLUDE    "M32DEF.INC"

      LDI   R20, 0x97
      LDI   R30, 0        ;number of 1s
      LDI   R16, 8        ;number of bits in a byte

AGAIN:
      ROR   R20           ;rotate right R20 and move LSB to C flag
      BRCC  NEXT          ;if C = 0 then go to NEXT
      INC   R30           ;increment R30
NEXT:
      DEC   R16           ;decrement R16
      BRNE  AGAIN         ;if R16 is not zero then go to AGAIN

      ROR   R20           ;one more time to leave R20 unchanged

HERE: JMP   HERE          ;stop here
```

</td><td>6</td><td>6</td></tr>
<tr><td>II.</td><td>4</td><td>


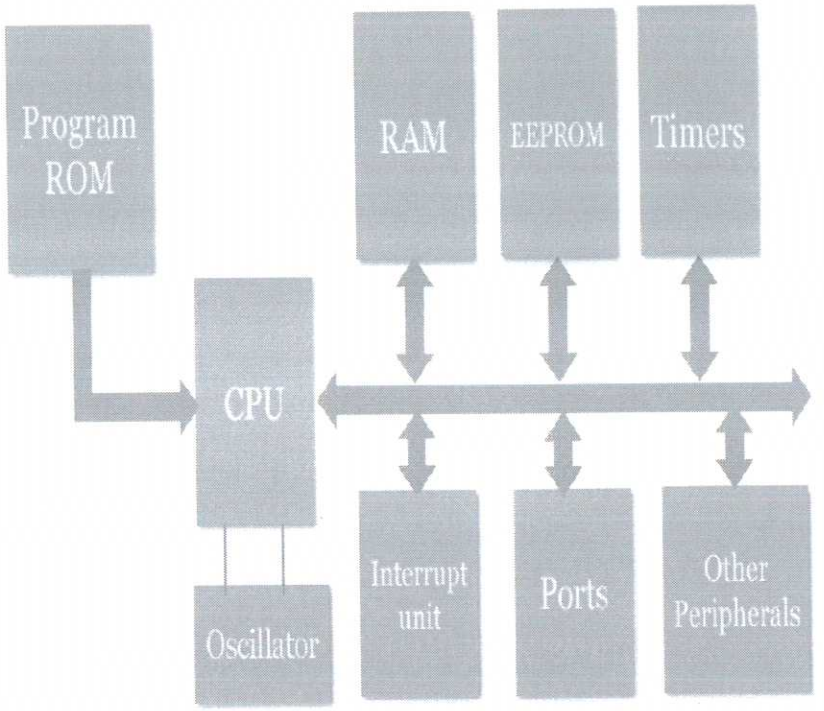
</td><td>6</td><td>6</td></tr>
<tr><td>II.</td><td>5</td><td>
The Serial Peripheral Interface (SPI) is a bus interface connection protocol originally started by Motorola Corp. It uses four pins for communication.

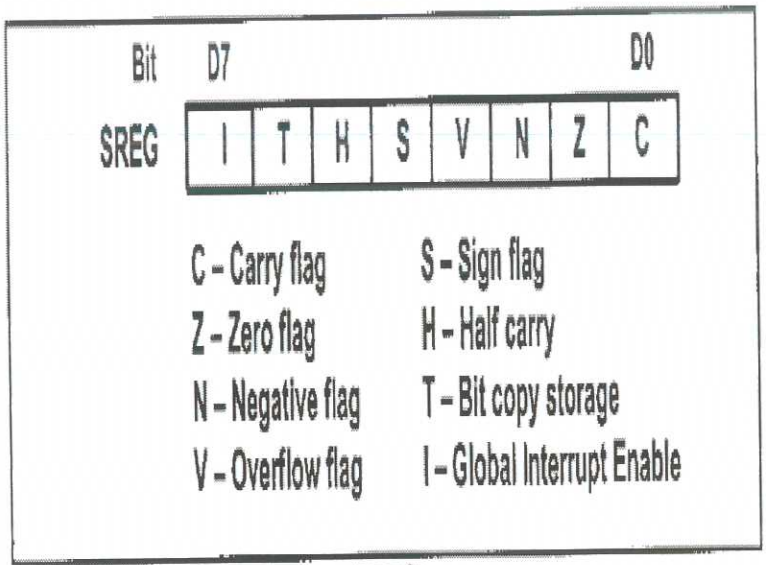
SDI (Serial Data Input)
SDO (Serial Data Output)
SCLK (Serial Clock)

</td><td>Fig 3
Expl 3</td><td>6</td></tr>
</table>

		• **CS** (Chip Select) It has two pins for data transfer called as SDI (Serial Data Input) and SDO (Serial Data Output). SCLK (Serial Clock) pin is used to synchronize data transfer and Master provides this clock. CS (Chip Select) pin is used by master to select slave device. SPI devices have 8-bit shift registers to send and receive data. Whenever master need to send data, it places data on shift register and generate required clock. Whenever master want to read data, slave places data on shift register and master generate required clock. Note that SPI is full duplex communication protocol i.e. data on master and slave shift registers get interchanged at a same time.  		
II.	6.	• Embedded NT • Application include internet Kiosk ,ATM. • Embedded NT works on 80x86 and pendium processor. • Windows XP Embedded • It requires large resources such as RAM and Flash Memory. • It is used in set top box etc • Embedded Linux • It is used in PDAs, Set Top Box, cellular phones etc •	2+2+2	6
II.	7.	• Embedded systems do a very specific task , they cannot be programmed to do different things. • Embedded systems have very limited resources, particularly the memory.	6	6

		Generally they don't have secondary storage devices such as the CDROM or the floppy disk. • Embedded systems have to work against some deadlines. • Embedded systems are uneasy for power. As many embedded systems operate through a battery ,the power consumption has to be very low. • Embedded systems used to be highly reliable. • Some embedded systems have to operate in extreme environmental conditions such as very high temperatures and humidity. • Embedded systems that address the consumer market are very cost - sensitive. pleted within a specific time.		
III.	1.	<u>Program ROM</u> • ROM is used to store program and is called program ROM • Program ROM size 1k to 256k • AVR was one of first microcontrollers to use on-chip Flash memory for program storage • For fast development flash memory is ideal as it can erase the data within seconds <u>Data RAM and EEPROM</u> • RAM space is for data storage • AVR has maximum of 64k of data RAM space • RAM spacehas three components • General purpose registers • I/O memory • Internal SRAM • 32 General purpose registers but the SRAM size and the I/O memory size varies from chip to chip <u>I/O pins</u> • 3 to 86 pins for I/O • Number of I/O depends on the number of pins in the package <u>Peripherals</u> • AVR comes with • ADC • Timers • USART	Fig 5 Expl 5	10



III.	2.	• C-Carry flag is set whenever there is a carry out from the D7 bit. • This flag bit is affected after an 8 bit addition or subtraction  Figure 2-8. Bits of Status Register (SREG) • Z-Zero flag reflect the result of an arithmetic or logic operation . • If the result is 0 then Z=1 • If the result is not = 0 then Z=0 • S-Binary representation of signed numbers uses D7 as the sign bit. • N-Negative flag reflect the result of an arithmetic operation . • If the D7 bit of the result is 0, then N=0 and the result is positive.	Fig 2 EXpl 3	5

		• If the D7 bit of the result is 1, then N=1 and the result is negative • V-Overflow flag is set whenever the result of an signed number operation is too large . • Overflow flag is used to detect errors in signed arithmetic operation • Sign flag is the result of exclusive ORing of N &V flags. • H-If there is a carry from D3 to D4 during ADD or SUB operation ,half carry bit is set, otherwise it is cleared. • This bit is used by instruction that performs BCD operation • The Global Interrupt Enable bit must be set (one) for the interrupts to be enabled • T-The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source and destination for the operated bit. • A bit from a register in the Register File can be cop-ied into T by the BST instruction and a bit in T can be copied into a bit in a register in the register file by the BLD instruction. •		
IV.	1.	Addressing Mode Overview o The CPU can access data in various way. The data could be: o In a register o Or in memory o Or provided as an immediate value. o These various ways of accessing data are called addressing mode. o. o There are 3 basic addressing modes for AVR; o Register addressing mode: one register, two register, immediate. o Direct addressing mode: o Indirect addressing mode: Single Register Addressing Mode o In this addressing, the operand is a register. That is only engage with one register. o The instructions can operates on any of the 32 registers (GPRs). COM R19,NEG R20 Two Register Addressing Mode o Two register addressing mode involves the use of two registers to hold the data to be manipulated. o The instructions can operates on any of the 32 registers (GPRs). o One of the register is the source register, Rs while the other one is destination register, Rd. ADD R20,R21;AND R30,R31 Immediate Addressing Mode o The constant value is sometimes referred to as immediate data since the operand comes immediately after the opcode. o Immediate addressing mode can be used to load data to any of the R16-R31 registers (GPRs). o The immediate addressing mode also used for arithmetic and logic instructions. i.e. LDI, ANDI, SUBI. Direct Addressing Mode o In direct addressing mode, the operand data is in a RAM memory location whose address is known, and this address is given as a part of the instruction. o The address field is a 16-bit address can take values from $0000- $FFFF.	4X2 (Any four)	8

		LDS R21,0X60,STS 0X61,R16 • I/O Direct Addressing Mode o The I/O direct addressing mode can address only the standard I/O registers. o The IN and OUT instructions use this addressing mode. o The address field is a 6-bit address can take values from $00-$3F. • IN R18,0X60,OUT 0X61,R20 Register Indirect Addressing Mode o In the register indirect addressing mode, a register is used as a pointer to the data memory location. o In AVR three registers are used as pointer register in which is made by combining two specific GPRs: o Register X Combining R26 and R27 o Register Y - Combining R28 and R29 o Register Z - Combining R30 and R31 Register Indirect Addressing Mode o The R26, R27, R28, R29, R30 and R31 GPRs can be referred to as XL, XH, YL, YH, ZL and ZH, respectively. Auto-increment & Auto-decrement Option for Pointer Registers o Because the pointer registers (X, Y and Z) are 16-bit registers, they can go from $0000 to $FFFF. o Using the INC ZL instruction to increment the pointer can cause a problem when an address such as $5FF is incremented due to the INC ZL will not propagate the carry into the ZH register. o So an auto-increment and auto-decrement for pointer registers were introduce to solve this problem. •		
IV.	2.	• In AVR there are two kinds of memory • Code memory space and data memory space • Data memory space is composed of three parts GPRs ,I/O memory and internal data SRAM GPR • GPRs use 32 byte of data memory space • They always take the address location $00-$1F • AVR have many registers for arithmetic and logic operation • CPU uses many registers to store data temporarily • That information can be byte of data to be processed or address pointing to the data to be fetched • AVR registers are 8 bit registers I/O memory (SFRs) • 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 • The AVR I/O memory is made of 8-bit registers • The number of location in the data memory for I/O depends on the pin numbers and peripheral functions supported by that chip • All AVRs have 64 bytes of I/O memory location • This 64 byte section is called standard I/O memory Internal data SRAM	Fig 3 Expl 4	7

- Internal data SRAM is widely used for storing data parameters by AVR programmers and compilers
- Generally it is called scratch pad
- Each location of SRAM can be accessed by its address
- Each location is 8 bit wide and can be used to store any data .

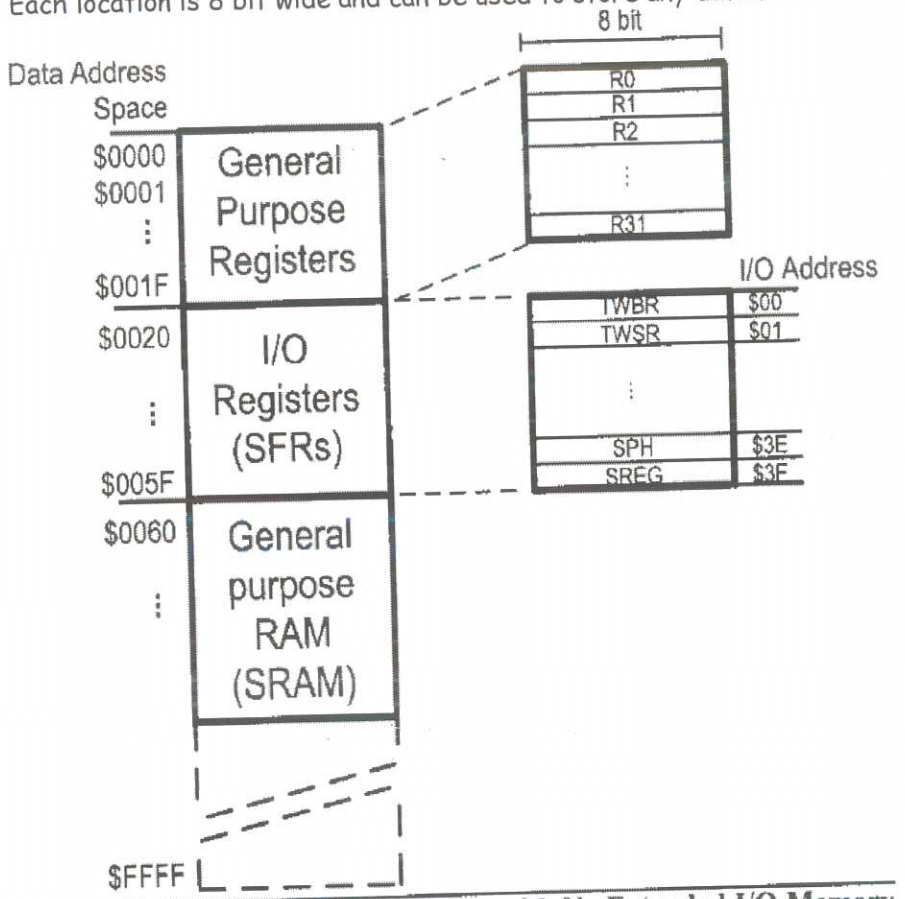
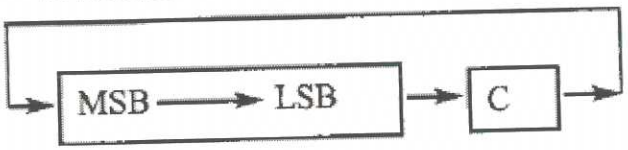


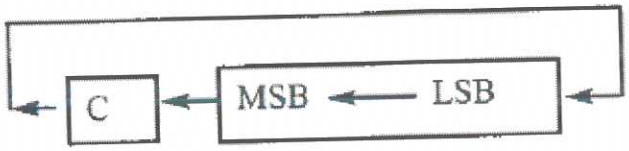
Figure 2-3. The Data Memory for AVRs with No Extended I/O Memory

V. 1. 2x5 10

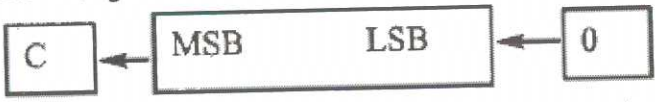
- For bit wise rotation we use ROR and ROL
- Rotating through the carry
- ROR
- ROR Rd; rotate Rd right through carry
- In ROR the C is moved to the MSB and the LSB is moved to the C



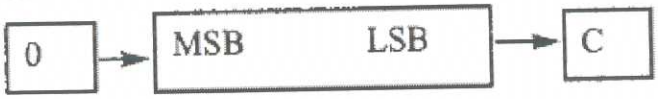
- In ROL the C is moved to the LSB and MSB is moved to the C



- LSL Instruction
- LSL Rd ;logical shift left



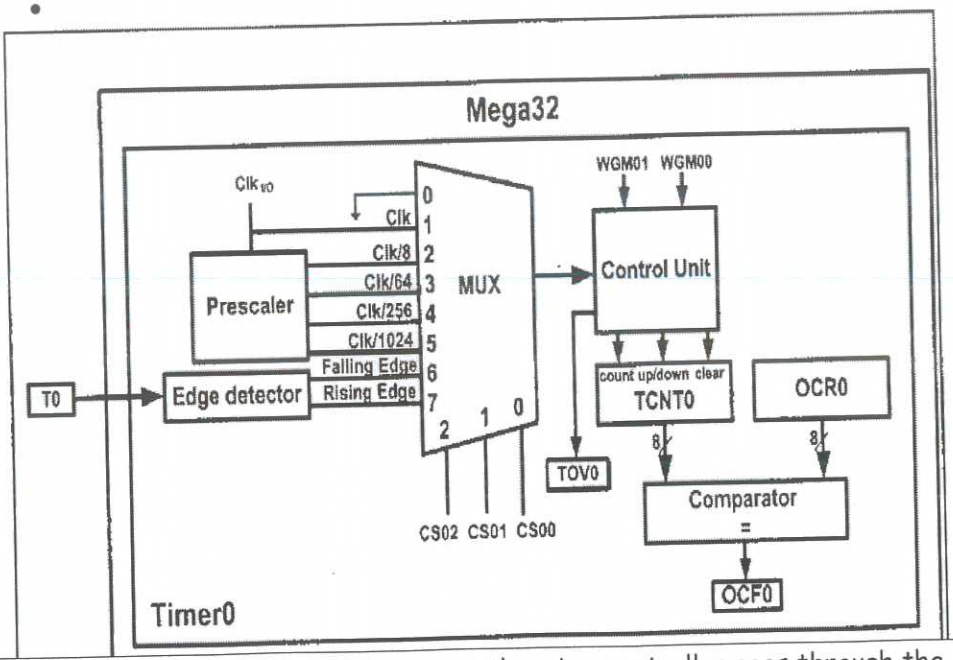
- In LSL 0 is moved to the LSB and the MSB is moved to the C flag

		• In LSR 0 moved to the MSB and the LSB is moved to the C flag  • • ASR means arithmetic shift left • The ASR can divide signed numbers into two • MSB not changed but is copied to D6,D6 is moves to D5,D5 to D4 and so on		

V	2.	LDS R1,0X300
		LDS R2,0X301
		LDS R3,0X302
		LDS R4,0X303
		ADD R1,R3
		ADD R4,R2
		STS 0X305,R1
		STS 0X306,R4
		HERE: JMP HERE
			5	5

VI.	1.	Both permit a group of instructions to be defined as a single entity with a unique given label or name called up when needed. •A subroutine is called by the BSR or JSR instructions, while a macro is called by simply using its name. •Simpler to write and use (subroutines are more complex, stacks are used) •Macros are faster than subroutines (no overheads, no saving of return addresse	5	5
VI	2.	*MOV* Rd, Rr • Moves (Copies) the value in register **Rr** to register **Rd**. • Allows any register to be copied to any other register. • Example: *MOV* r1, r10 *LDS* Rd, K • Loads (Copies) the value in memory location **K** into register **Rd**. • **Rd** may be any register. • **K** may be any address between 0 and 65535. • This is known as **Direct** addressing. • Often use a variable to represent the memory address. o **R1** receives the value stored in in **R10**. *LD* Rd, X+ • Loads (Copies) the value in memory location stored in the **X** register (**R27:R26**) into register **Rd** and increments the value in the **X** register. • **X** is incremented after the copy. • This is called post-increment. *STS* K, Rr • Stores (Copies) the value in register **Rr** into memory location **k**.	2.5x4 (Any four)	10

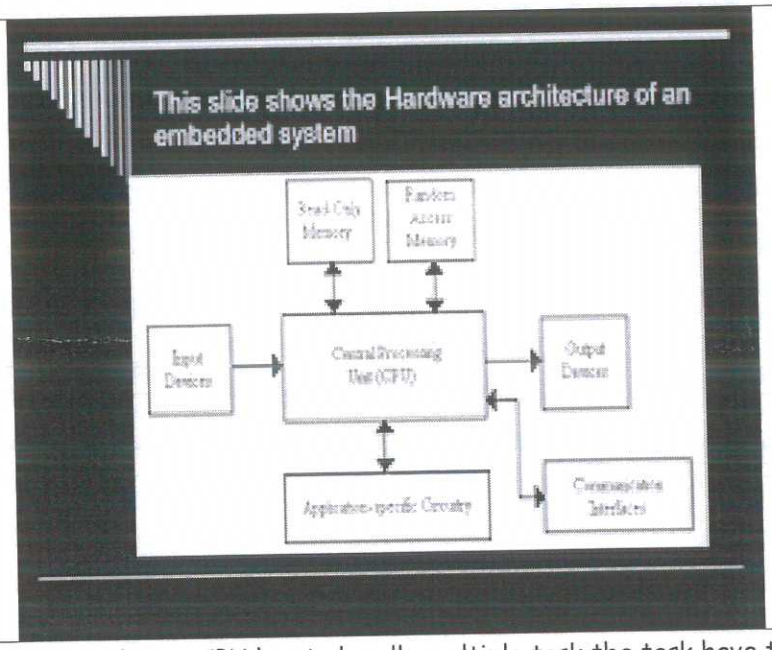
		• **Rr** may be any register. • **K** may be any address between 0 and 65535. • This is similar to *LDS*. *ST* X, Rr • Stores (Copies) the value in register **Rr** into the memory location stored in the **X** register (**R27:R26**). • This is known as **Indirect** addressing. • Post-increment and pre-decrement instructions similar to *LD* are available: o *ST* X+, Rr o *ST* -X, Rr		
VII.	1.	Unsigned char • The unsigned char is an 8 bit data type that takes a value in the range of 0-255 (00-FFH). • It is one of the most widely used data types for the AVR. • In many situations ,such as setting a counter value, where there is no need for signed data, we should use the unsigned char instead of the signed char. • Because the AVR microcontroller has a limited number of registers and data RAM locations, using int in place of char can lead to the need for more memory space. Signed char • The signed char is an 8 bit type that uses the most significant bit (D7 of D7-D0) to represent the – or + value. • As a result, we have only 7 bits for the magnitude of the signed number, giving us values from -128 to +127. • In situations where + and – are needed to represent a given quantity such as temperature ,the use of the signed char data type is necessary. If we don't use the keyword unsigned, the default is the signed value Unsigned int • The unsigned int is a 16 bit data type that takes a value in the range of 0 to 65,535 (0000-FFFFH). • In the AVR, unsigned int is used to define 16 bit variables such as memory addresses. • It is also used to set counter values for more than256. • Because the AVR is an 8 bit microcontroller and the int data type takes 2 bytes of RAM, we must not use the int data type unless we have to. • Because registers and memory accesses are in 8 bit chunks, the misuse of int variable will result in larger hex files, slower execution of program, and more memory usage. Signed int • Signed int is a 16 bit data type that uses the most significant bit (D15 of D15-D0) to represent the – or +value.	7	7

		• As a result, we have only 15 bits for the magnitude of the number, or values from -32,768 to +32,767.		
VII.	2.	• In AVR, for each of the timers, there is a TCNTn (timer/counter) register. • That means in AT mega 32 we have TCNT0, TCNT1, and TCNT2 . • The TCNTn register is a counter. • Upon register, the TCNTn contains 0. • It counts up with each pulse. • The contents of the timers/counters can be accessed using the TCNTn. • Each timer has a TOVn (Timer Overflow) flag. • When a timer overflows, its TOVn flag will be set. • Each timer also has the TCCRn (timer/ counter control register) register for setting of operation. • We can specify Timer0 to work as a timer or a counter by loading proper values into the TCCR0. • Each timer also has an OCRn (Output Compare Register) register. • The content of the OCRn is compared with the content of the TCNTn. • When they are in equal the OCFn (Output Compare Flag) flag will setWe can load a value into the TCNTn register or read its value. • The timer registers are located in the I/O register memory. • Therefore, we can read or write from timer registers using IN and OUT instructions, like other I/O registers • 	Fig 4 Expl 4	8
VIII	1.	• Upon activation of an interrupt, the microcontroller goes through the following steps:- • 1.it finishes the instruction it is currently executing and saves the addresses of the next instruction (program counter on the stack. • 2.It jumps to a fixed location in memory called the *interrupt vector table.* • The interrupt vector table directs the microcontroller to the address of the interrupt service routine (ISR). • 3.The microcontroller starts to execute the interrupt service subroutine until it reaches the last instruction of the subroutine, which is RETI	5	5

		(Return from interrupt). • 4.upon executing the RETI instruction, the microcontroller returns to the place where it was interrupted . • First it gets the program counter (PC) address from the stack by popping the top bytes of the stack into the PC. • Then it starts to execute from that address.		

VIII.	2.	
		
		`#include <avr/io.h>` `//standard AVR header`
		`int main(void)`
		`{`
		`unsigned char x, y;`
		`unsigned char mybyte = 0x29;`
		
		`DDRB = DDRC = 0xFF;` `//make Ports B and C output`
		`x = mybyte & 0x0F;` `//mask upper 4 bits`
		`PORTB = x | 0x30;` `//make it ASCII`
		`y = mybyte & 0xF0;` `//mask lower 4 bits`
		`y = y >> 4;` `//shift it to lower 4 bits`
		`PORTC = y | 0x30;` `//make it ASCII`
		
		`return 0;`
		`}`
			10	10

IX.	1.	▸ Central Processing Unit: The central Processing Unit can be any of the following: micro-controller, microprocessor or Digital Signal Processor .A micro- controller is a low-cost processor. ▸ Memory: The memory is categorized as Random Access Memory and Read Only Memory. ▸ Input Devices: Many embedded sysytems will have a small key pad -you press one key to give a specific command .A key pad may be used to in put only the digits. ▸ Output Devices: The output devices of **the** embedded systems also have very limited capability .Some embedded systems will have a few Light Emitting Diodes to indicate the health status of the system modules ,or for visual indication of alarms. ▸ Communication Interfaces: The embedded systems may need to interact with other embedded systems or they may have to transmit data to a desktop .To facilitate this, the embedded systems are provided with one or a few communication interfaces such as RS232, RS422, RS485 ,Universal Serial Bus, IEEE 1394,Ethernet etc. ▸ Application-specific circuitry: Sensors, transducers, special processing and control circuitry may be required for an embedded system ,depending on its application .This circuitry interacts with the processor to carry out the necessary work	Fig 5 Expl 5	10

				
IX.	2.	■ Since only one CPU has to handle multiple task the task have to share the CPU time in disciplined way so that one task does not get lot of time while others wait for long time . A mechanism for deciding which task will get the CPU time next is to be worked out. This known as task scheduling ■ Different task may have to share same resources. ■ Two task have to share a printer. ■ Resources shared by two or more tasks are called shared resources. ■ Tasks should maintain discipline to share resources ■ Ensuring that two or more tasks access a shared resources without corrupting data is called mutual exclution	Fig 3 Expl 6	9
X.	1.	1. Consumer appliances 2. Industrial automation 3. Medical electronics 4. Computer networking 5. Telecommunications 6. Wireless technologies 7. Instrumentation 8. Security	5	5
X.	2.	★ **Kernel** :Kernel manages tasks to achieve the desired performance of the embedded systems . ★ To manage the tasks , the important requirements are to schedule the tasks and to provide inter-task communication facilities. ★ To achieve these two requirements, kernel objects are defined such as tasks ,mutexes , ISRs , events ,message boxes, mailboxes, pipes and timers. Kernel provides the memory management services, time management services, interrupt handling services and device management services. ❑ **Device Manager** :The I/O devices are used to send /receive data from	Fig 5 Expl 5	10

the embedded systems.

- The OS manages the I/O device through interrupts and device drivers.
- Device drivers provide the necessary interface between the application and hardware.
- **Communication protocol software:** If the embedded systems has communication interfaces such as Ethernet , USB etc, the upper layer protocols such as TCP/IP stack need to be integrated with the OS. Then **Libraries :** The operating system may have some C/C++ library files in object code ,which can be used through the API calls.

- **File system :** Most of the embedded systems don't have a secondary storage. In suc cases, the ROM is used to store the program .
- In case a file system is required , a small file system can be developed a Flash memory .
- Some embedded systems may use a secondary storage just for booting
- For example ,an Internet kiosk may use a CDROM for initial bootup.
- the embedded systems can be networked -enable.