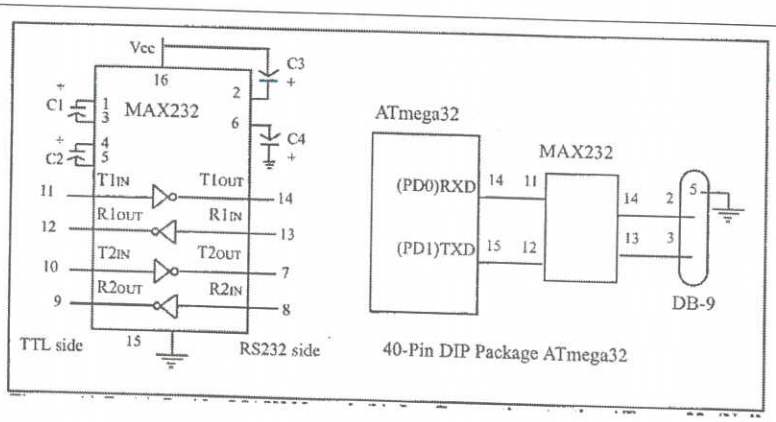
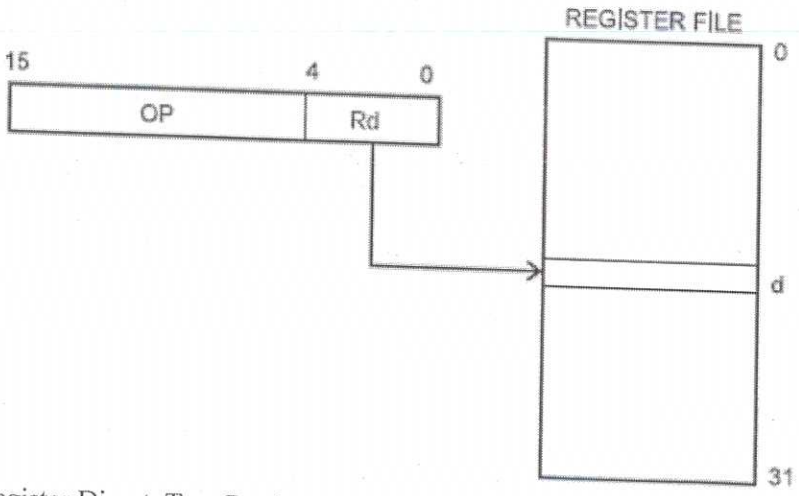
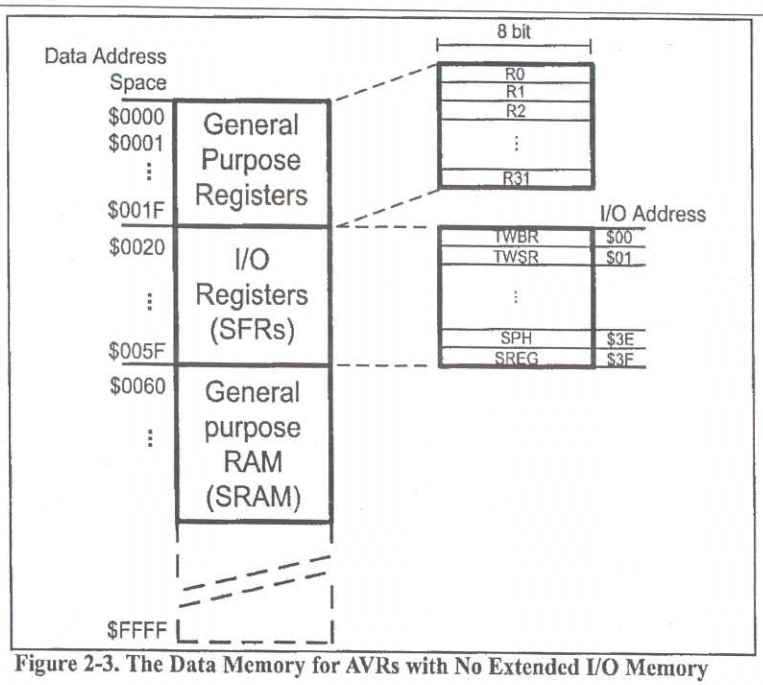
Scheme of Evaluation

Scoring Indicators.

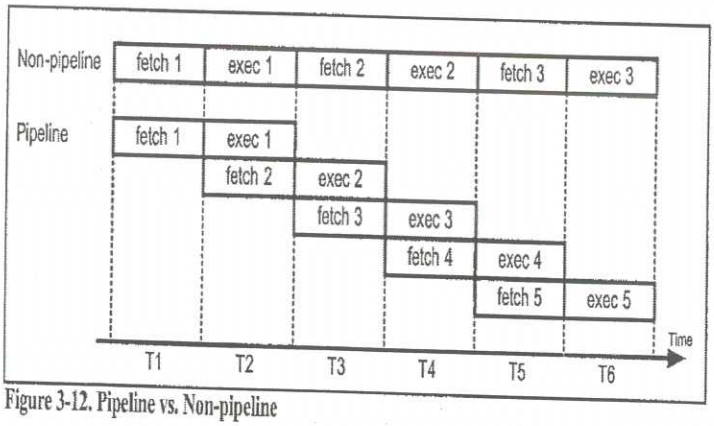
Revision 2015 Embedded Systems 5041				
Quest No:	Scoring Indicator	Split up score	Sub Total	Total
1.	.EQU, .SET, .ORG,.INCLUDE	2	2	2
2.	64 Byte, 32 KB	2	2	2
3.	AND Rd, Rr OR Rd,Rr EOR Rd,Rr	2	2	2
4.	0x05, 0x2C	2	2	2
5.	VxWorks, QNX, eCos, RTLinux	2	2	2
II 1.	The processor saves the address of the next instruction in the stack and loads the PC with the starting address of the subroutine. Thus the processor transfers control to the sub-routine. On executing the RET instruction at the end of the subroutine, the top of the stack is retrieved and the PC is loaded with the top of the stack and the execution is resumed.	6	6	6
2	Tiny AVR- Less memory, small size, suitable for small applications Mega AVR – Memory -256 KB, higher number of inbuilt peripherals , suitable for advanced applications Xmega AVR – high speed and program memory, used for complex applications.	3*2	6	6
3.	There are 4 ports A,BC and D in Atmega 32. Each port is associated with three I/O registers- DDRx, PINx and PORTx. DDRx is used for making PORTx input or output. For making the port , an output port, DDRx is loaded with $FF. For making it as input port, DDRx is loaded with $00. PINx is used for inputting the data. Ie, the data has to be placed in the PINx register, if it needs to be read by the processor. PORTx is used for outputting the data.	6	6	6

				
6.	Context Switching involves storing the context or state of a process so that it can be reloaded when required and execution can be resumed from the same point as earlier. This is a feature of a multitasking operating system and allows a single CPU to be shared by multiple processes. The illusion that all the tasks are running concurrently is achieved by allowing each to have a share of the processor time. This is the core functionality of a kernel. The way that time is allocated between tasks is termed "scheduling". The scheduler is the software that determines which task should be run next. The logic of the scheduler and the mechanism that determines when it should be run is the scheduling algorithm.	3+3	6	6
7.	EEPROM memory is used for for storing data which is not changed frequently in AVR. EEPROM does not lose its data when the power is off, where as SRAM does. Therefore EEPROM is used for storing data and parameters that are rarely changed, while SRAM is used for storing data and parameters that are changed frequently.	2 + 2*2	6	6
IIIa.	Register Direct, Single Register Rd Eg. Inc R16  Register Direct, Two Registers Rd and Rr Add r16,r17	Any four 2*4	8	8

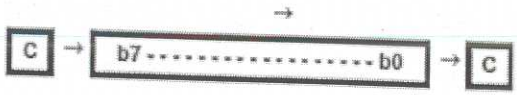
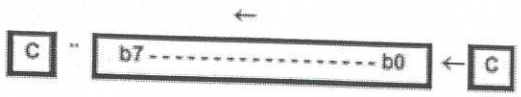
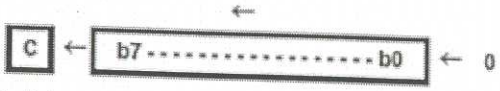
2

IV a.	 **Figure 2-3. The Data Memory for AVRs with No Extended I/O Memory** General purpose registers: - 32 in number, using 32 bytes of memory, taking the address locations from $00-$FF. I/O memory (SFR) : dedicated to specific functions such as status register, timers serial communication etc. All of AVR chipsets have at least 64 bytes of I/O memory , which is called the standard I/O memory. Internal Data SRAM :- Scratch pad memory for storing data by programmers. In addition to SRAM, AVR has EEPROM memory for storing data . The advantage of EEPROM is that the data is not lost even when the power is off.	4+4	8	8
b.	AVR is a 8-AVR is a 8-bit RISC architecture (Reduced Instruction Set Computing) microcontroller bit RISC architecture (Reduced Instruction Set Computing) microcontroller on-chip programmable flash memory, SRAM, IO data space & EEPROM. AVR is the first MCU in market which has on-chip flash storage.	3+4	7	7

3

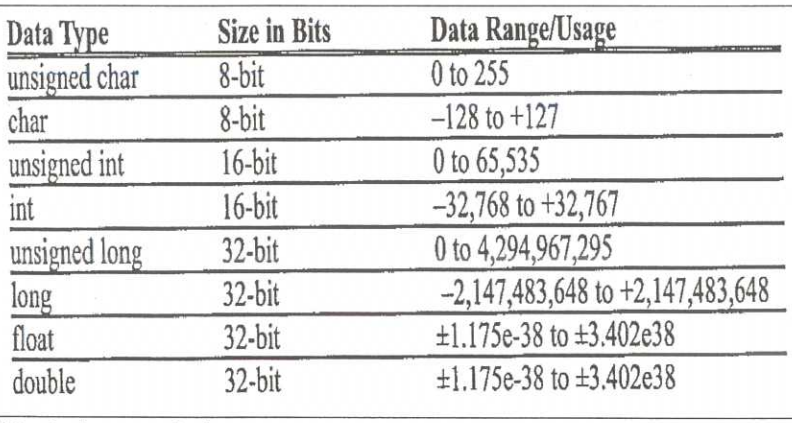
	their instruction throughput AVR has a two-stage instruction pipeline. While one instruction is executed, the next is fetched from the program memory.  Figure 3-12. Pipeline vs. Non-pipeline			

b	.INCLUDE "M32DEF.INC"
	.ORG 0
	
	LDI R16, $AA ; load the value $AA
	OUT PORTA, R16 ; output the value to PORTA
	LDI R20, 200 ; load the value 200
	L1: COM R16 ; complement
	OUT PORTA
	DEC R20
	BRNE L1 ; looping 200 times
		7	7	7

VI a	Two Rotate Instructions: ROR Rd ; rotate right through carry  ROL Rd ; Rotate left through carry  Shift Instructions: LSL Rd Logical Shift left, Arithmetic Shift left  LSR Rd; Logical Shift Right	2*4 = 8	8	8

4

b	**Steps to program Timer0 in Normal mode**
	
	To generate a time delay using Timer0 in Normal mode, the following steps are taken:
	
	1. Load the TCNT0 register with the initial count value.
	2. Load the value into the TCCR0 register, indicating which mode (8-bit or 16-bit) is to be used and the prescaler option. When you select the clock source, the timer/counter starts to count, and each tick causes the content of the timer/counter to increment by 1.
	3. Keep monitoring the timer overflow flag (TOV0) to see if it is raised. Get out of the loop when TOV0 becomes high.
	4. Stop the timer by disconnecting the clock source, using the following instructions:
	`LDI R20,0x00`
	`OUT TCCR0,R20 ;timer stopped, mode=Normal`
	
	5. Clear the TOV0 flag for the next round.
	6. Go back to Step 1 to load TCNT0 again.
		7	7	7

VIII a	 **Data Type** / **Size in Bits** / **Data Range/Usage** unsigned char — 8-bit — 0 to 255 char — 8-bit — –128 to +127 unsigned int — 16-bit — 0 to 65,535 int — 16-bit — –32,768 to +32,767 unsigned long — 32-bit — 0 to 4,294,967,295 long — 32-bit — –2,147,483,648 to +2,147,483,648 float — 32-bit — ±1.175e-38 to ±3.402e38 double — 32-bit — ±1.175e-38 to ±3.402e38	4*2	8	8

b	`# include <avr/io.h>`
	`# include <avr/delay.h>`
	
	`int main (void)`
	`{`
	`DDRB = 0xFF;  // setting PORTB as output port`
	`While(1)`
	`{`
	`PORTB = PORTB|000001000 ; // setting the fourth pin`
	`_delay_ms(1000);  // delay of 1 sec`
	`PORTB = PORTB|111110111; // resetting the fourth pin`
	`}`
	`Return 0;`
	`}`
		7	7	7

5

	• Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. • Open source and extensible hardware - experienced circuit designers can make their own version of the module, extending it and improving it.			
Xa	a) Embedded systems are application specific & single functioned; the application is known apriori, the programs are executed repeatedly. b) Efficiency is of paramount importance for embedded systems. They are optimized for energy, code size, execution time, weight & dimensions, and cost. c) Embedded systems are typically designed to meet real-time constraints; a real-time system reacts to stimuli from the controlled object/ operator within the time interval dictated by the environment. For real-time systems, right answers arriving too late (or even too early) are wrong. d) Embedded systems often interact (sense, manipulate & communicate) with the external world through sensors and actuators and hence are typically reactive systems; a reactive system is in continual interaction with the environment and executes at a pace determined by that environment. e) They generally have minimal or no user interface.	8	8	8
b	An embedded system has 3 components: • It has the hardware. • It has software program. • It has an actual real-time operating system (RTOS) that supervises the utility software and offer a mechanism to let the processor run a process as in step with scheduling by means of following a plan to manipulate the latencies. RTOS defines the manner the system works. It units the rules throughout the execution of application software. A small scale embedded device won't have RTOS. The embedded hardware primarily includes the processor, memory, bus, peripheral devices, I/O ports, and various controllers. The embedded software usually contains the embedded operating system and various applications.	4+3	7	7

6