Set-02

DECEMBER 2018 / END-SEM

F. Y. M. TECH. (E&TC) (SEMESTER - I)

COURSE NAME: Advanced Embedded Processors and Programming

COURSE CODE: ETPA11182

(PATTERN 2018)

Scheme of Marking/Solution Model Answer

Q.1) a) Non-RISC features of RISC architecture are as follows:

1. Control over ALU & Shifter operand
2. Auto-increment and auto-decrement addressing mode
3. Load and store multiple instructions
4. Conditional execution of almost every instruction
5. High Code density.(Less memory)
6. Hardware Debug Technology
7. Orthogonal instruction Set

[Any three features with explanation 01 marks each]

OR

b) Selection criterions of memory in embedded applications is as follows:

1. View point of Hardware designer
2. Viewpoint of Software designer
3. Serial access
4. Sequential access
5. Memory expansion scope

[Any three criteria with explanation 01 marks each]

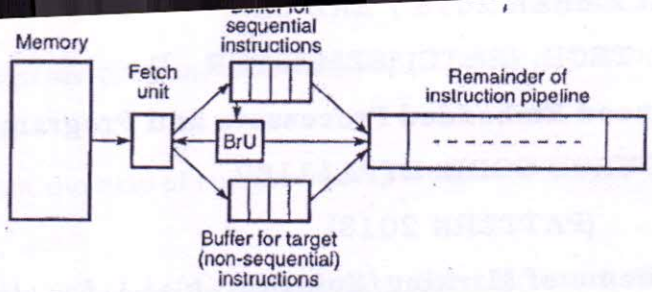
Q.2) a) typical classification of processor architecture is based on modes of data operated for Instruction as

1. Single Instruction single data (SISD)
2. Single Instruction Multiple data (SIMD)
3. Multiple Instruction single data (MISD)
4. Multiple Instruction Multiple data (MIMD)

[3 marks]

OR

b) Method to reduce data hazards in pipeline architecture is operand forwarding. In the same clock of pipeline stage, execution and write back register phase is active. Output from execution unit is forwarded well in advance to be used by write back stage. FIFO is used for the same purpose as follows:



[3 marks]

Q.3) a) Cortex architecture is popular in embedded systems because:

1. Greater performance efficiency
2. Low power consumption
3. Enhanced determinism
4. Improved code density
5. Ease of use
6. Lower cost solutions
7. Wide choice of development tools
8. Nonmaskable interrupts for critical tasks
9. highly deterministic nested vector interrupts

[Any four points 1/2 mark each]

OR

b) A-R-M profile in cortex is as follows:

- *A Profile (ARMv7-A)*: Application processors which are designed to handle complex applications such as high-end embedded operating systems (OSs) (e.g., Symbian, Linux, and Windows Embedded). These processors requiring the highest processing power, virtual memory system support with memory management units (MMUs), and, optionally, enhanced Java support and a secure program execution environment. Example products include high-end mobile phones and electronic wallets for financial transactions.
- *R Profile (ARMv7-R)*: Real-time, high-performance processors targeted primarily at the higher end of the real-time[1] market—those applications, such as high-end breaking systems and hard drive controllers, in which high processing power and high reliability are essential and for which low latency is important.
- *M Profile (ARMv7-M)*: Processors targeting low-cost applications in which processing efficiency is important and cost, power consumption, low interrupt latency, and ease of use are critical, as well as industrial control applications, including real-time control systems.

[2 marks]

Q.4) a) RTOS requirement in embedded application is because of the need for :

1. Integration of application Specific Services
2. Secured/Protected application code with OS code
3. Control by an application rather than OS
4. Scalability requirement in an application
5. Service times needs to more reliable and accurate

6. Customization should be facilitated

[02marks]

Alternative software architectures used in embedded systems with pros and cons are summarized as

Round Robin- Features

1. Simple Architecture.
2. No Interrupts
3. No Shared data
4. No latency concerns

Ex. Digital Multimeter

Round Robin with Interrupts- Features

1. It's more sophisticated software architecture.
2. Interrupt routines deal with very urgent needs of hardware.
3. Little bit more control over priorities.
4. Interrupt routines can get very good response.
5. Shared data must be taken care of

Function Queue Scheduling

1. In this architecture the interrupt routines add function pointer to a queue of function pointers for main function to call.
2. The main routine reads the pointers and calls the functions.
3. Main function can call the functions in any order, depending upon the priority.
4. This requires clever coding in the routines that queue up the function pointers.

- Real Time Operating System (RTOS)
  1. Very stable response time
  2. Zero task interrupt latency
  3. Highly Scalable
  4. Complex design can be granulize with concurrent task model

[1 and 1/2 marks each]

Q.4) b) Requirements of architecture for porting operating system onto it are:

1. You must have a C compiler for the processor and the C compiler must be able to produce reentrant code.
2. You must be able to disable and enable interrupts from C.
3. The processor must support interrupts and you need to provide an interrupt that occurs at regular intervals (typically between 10 to 100 Hz).
4. The processor must support a hardware stack, and the processor must be able to store a fair amount of data on the stack (possibly many Kbytes).
5. The processor must have instructions to load and store the stack pointer and other CPU registers either on the stack or in memory.
6. Stack manipulation and memory endianess configuration should be facilitated

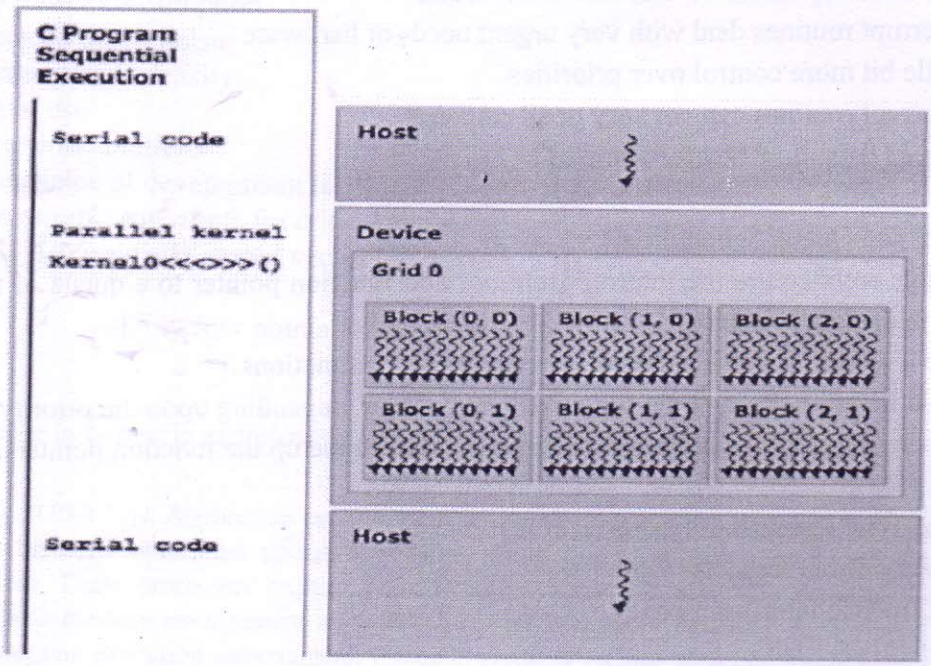
[01 mark each]

OR

Q.5) a) Different types of RTOS are:

1. Monolithic : services are integrated in kernel , less customization, no scalability, complex for debugging, no ROMable etc..
   e.g. VxWorks
2. Microkernel : services can be added to kernel. Minimalistic kernel, highly scalable ,fine granular etc
   E,g QNX
3. Decoupled : separate real time kernel that runs as highest priority thread to non real time kernel.
   E,g RTLinux
   [Detailed explanation of each 2+2+4 marks]

Q.5) b) CUDA programming model:



[03 marks]

Example of vector addition :

```
__global__ void  vec_add (float *A, float *B, float *C, int N)
{
        // Using 1D threadID and block ID
        int i = threadIdx.x + blockDim.x*blockIdx.x;

        if (i>=N) {return;}

        C[i] = A[i] + B[i];
}

int main()
{
         ....
         // Launch kernel with N/256 blocks of 256 threads
         int blocks = int(N-0.5)/256 + 1;
         vec_add<<<blocks, 256>>> (A_d, B_d, C_d, N);
}
```

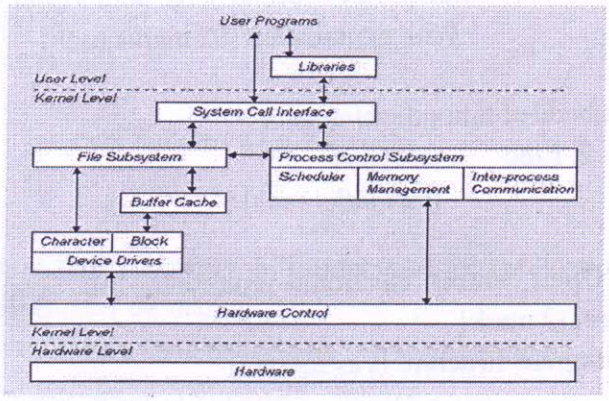
[03 marks]

Q. 6) a) Device driver:

- An OS component that is responsible for hiding the complexity of an I/O device, so that the OS can access various devices in a uniform manner. A set of routines that communicate with a hardware device and provide a uniform interface to the operating system kernel.
- A self-contained component that can be added to, or removed from, the operating system dynamically.
- Management of data flow and control between user programs and a peripheral device.

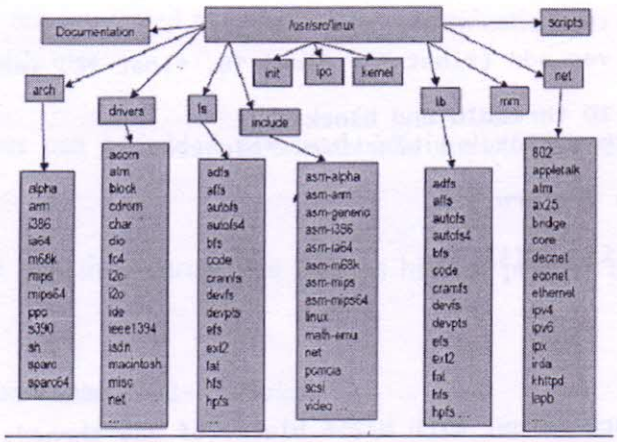
[02 marks]

Typical structure of device driver:



With explanation [06 marks]

Q. 6) b) Different file systems used in embedded Linux:



[02 marks]

- Storage-based file systems
- The **ext2** filesystem is the canonical Linux filesystem
- **ext3** adds journaling capabilities to protect the integrity of the file system and updates to it
- **ext4** is also a journaling filesystem, but adds new allocation methods that improve efficiency and performance when allocating large amounts of storage

other types of Linux file systems such as **JFS**, **Reiser FS**, and **XFS**, are designed and optimized for use on traditional storage media such as hard drives

[04 marks]

OR

Q.7) a) steps of porting linux on ARM architecture are as follows :

1. Configuration and compilation of Bootloader
2. Configuration and compilation of Kernel
3. Configuration and compilation of File system
4. Configuration and compilation of second stage boot loader

With explanation [02 marks each]

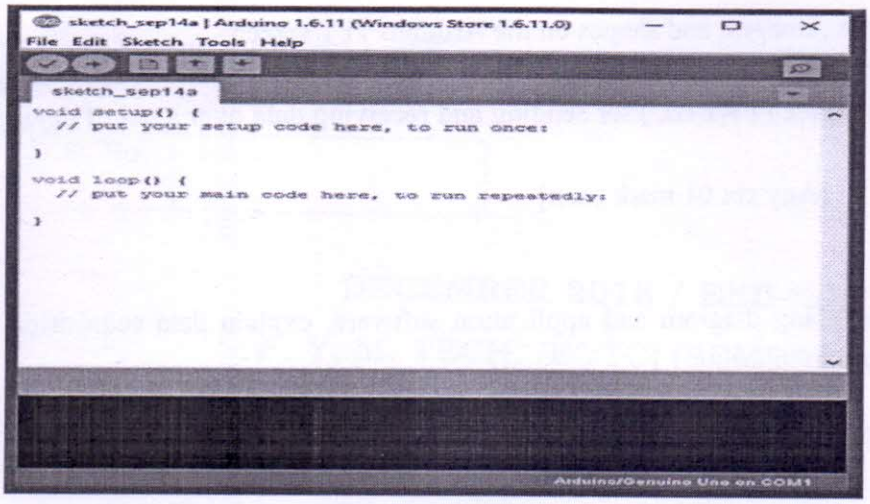
Q.7) b) utilities in embedded linux development

1. Minicom 2. Libc 3. Busybox

[02 marks each]

Q.8) a) Arduino is **Open Source** electronic prototyping **platform** based on flexible **easy to use** hardware and software. [2 mark]

typical application program structure is as follows:



Explanation of typical structure [6 marks]

Q.8) b) features of Arduino UNO board are:

1. Microcontroller: ATmega328
2. Operating Voltage: 5V
3. Input Voltage (recommended): 7-12V
4. Input Voltage (limits): 6-20V
5. Digital I/O Pins: 14 (of which 6 provide PWM output)
6. Analog Input Pins: 6
7. DC Current per I/O Pin: 40 mA
8. DC Current for 3.3V Pin: 50 mA
9. Flash Memory: 32 KB of which 0.5 KB used by bootloader
10. SRAM: 2 KB (ATmega328)
11. EEPROM: 1 KB (ATmega328)
12. Clock Speed: 16 MHz

[any six features 01 mark each]

OR

Q.9) a) significance of Arduino Library is ready API to be used for application development without knowing greater depth of architecture. [02 marks]

Various components in standard library used for application development:

- **EEPROM** - reading and writing to "permanent" storage
- **Ethernet** / Ethernet 2 –
- **Firmata** - communicating with applications on computer by standard serial protocol.
- **GSM** - for connecting to a GSM/GRPS network with the GSM shield.
- **LiquidCrystal** - for controlling liquid crystal displays (LCDs)
- **SD** - for reading and writing SD cards
- **Servo** - for controlling servo motors
- **SPI** - for communicating with devices using the Serial Peripheral Interface (SPI) Bus
- **SoftwareSerial** - for serial communication on any digital pins. Version 1.0 and later of Arduino incorporate Mikal Hart's NewSoftSerial library as SoftwareSerial.
- **Stepper** - for controlling stepper motors

- **TFT** - for drawing text , images, and shapes on the Arduino TFT screen
- **WiFi** - for connecting to the internet using the Arduino WiFi shield
- **Wire** - Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensor

[Any six 01 mark each]

Q.9) b) with hardware interfacing diagram and application software, explain data acquisition and control system for temperature monitoring

Hardware interfacing diagram [03 marks]
Algorithm and program [03 marks]