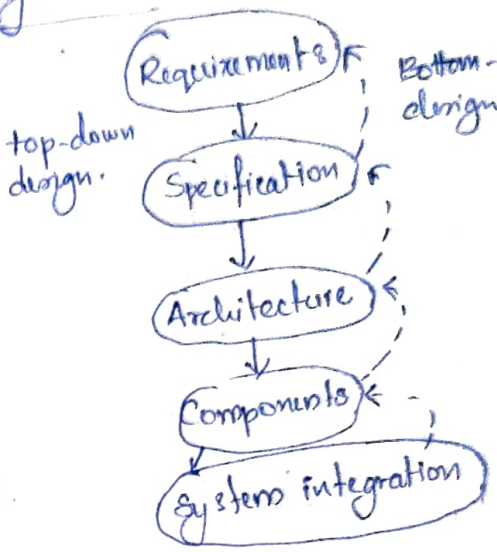
01/07/2021
Thursday.

SJC17CS035.
Christy Varghese.

Embedded system.
CS 404.

① (a) Steps of Embedded system Design Process.



Step 1: Top-down Design

The most abstract description of the system and conclude with concrete details. Bottom-up designs steps are shown in the figure as dashed-line arrows. During the design process we have to consider the major goals of the design such as

- Manufacturing cost
- Performance
- Power consumption.

Step 2: Requirements

Generally requirement proceeds in two phase:

(i) First gather an informal description from the customers known as requirement.

(ii) Secondly, refine the requirement into a specification that contains enough information to begin designing the system architecture.

Requirements may be functional or non-functional. Typical non-functional requirements include:

→ Performance
→ Cost
→ Physical weight & size.

christy

Step 3. Design the system architecture.

The architecture of an embedded system depends on:

(i) whether the system is real time.
(ii) whether OS needs to be embedded.
(iii) Cost, size, power consumption etc.

Step 4: Selet the OS

If operating system we can selet,
(i) Real time OS like RTLinux, VRTX etc.
(ii) Nonreal OS systems like windows. etc.

Step 5: Choose the development platfoams.

The development platforms of an embedded system includes the following:

(i) The hardware platforms.
(ii) Operating system.
(iii) The development tools.
(iv) The programming language

Step 6: Choosing the H/w components

The component effort builds those components in conformance to the architecture and specification. The components will in general include both h/w & s/w modules

Step 7: Designing H/w & s/w comp.

The component design effort build those components in conformance.

Step 8: System Integration

Only after the components are built does one have the satisfaction of putting them together and seeing a working system. Bugs are typically found during system integration and good planing can help us find the bugs quickly.

chhisty

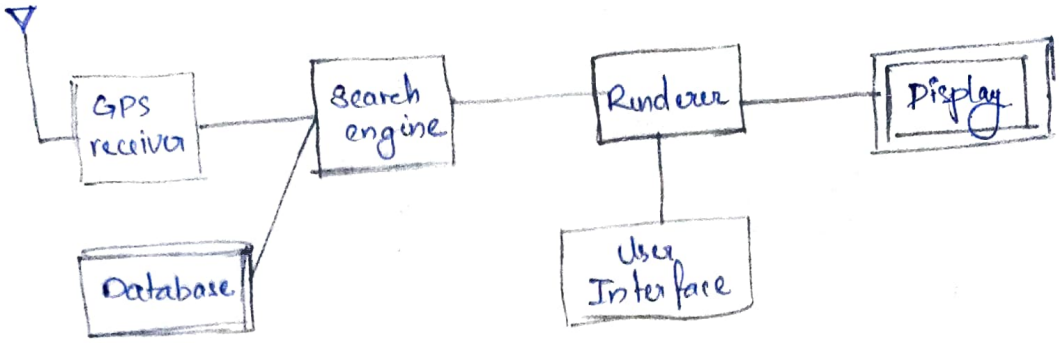
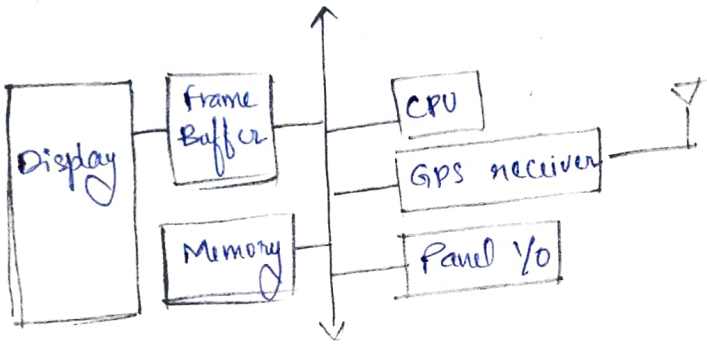
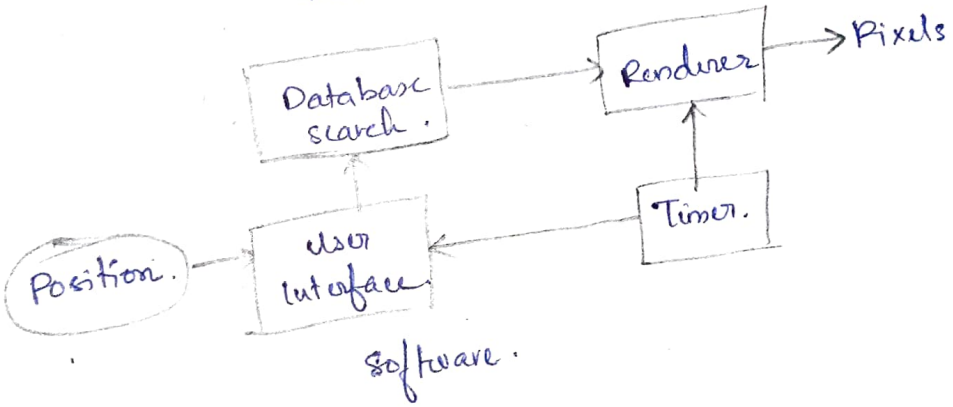


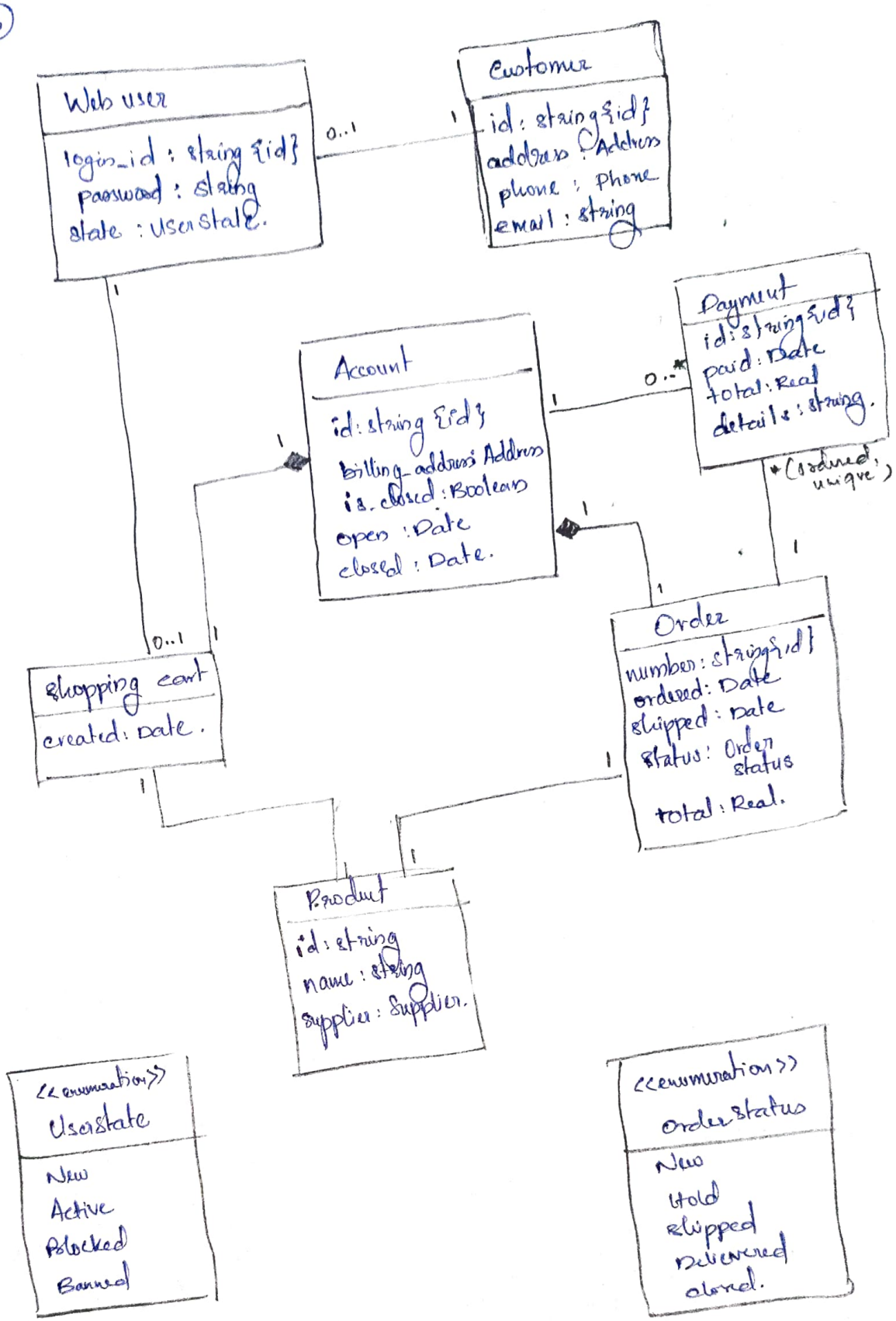
Fig. Block diagram for the movingmap





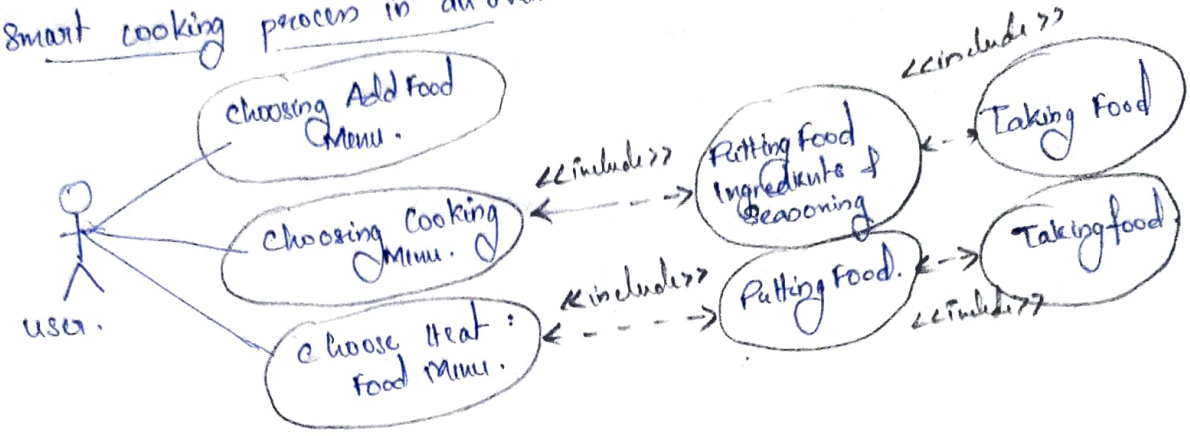
(4)

(1)(b)


Web user
login_id : string {id}
password : string
state : UserState.
0..1
1
Customer
id : string {id}
address : Address
phone : Phone
email : string
1
Account
id : string {id}
billing_address : Address
is_closed : Boolean
open : Date
closed : Date.
1
0..*
Payment
id : string {id}
paid : Date
total : Real
details : string.
*{ordered, unique}
1
0..1
1
Shopping cart
created : Date.
1
1
Order
number : string {id}
ordered : Date
shipped : Date
status : Order status
total : Real.
1
1
Product
id : string
name : string
supplier : Supplier.
<<enumeration>>
UserState
New
Active
Blocked
Banned
<<enumeration>>
OrderStatus
New
Hold
Shipped
Delivered
closed.

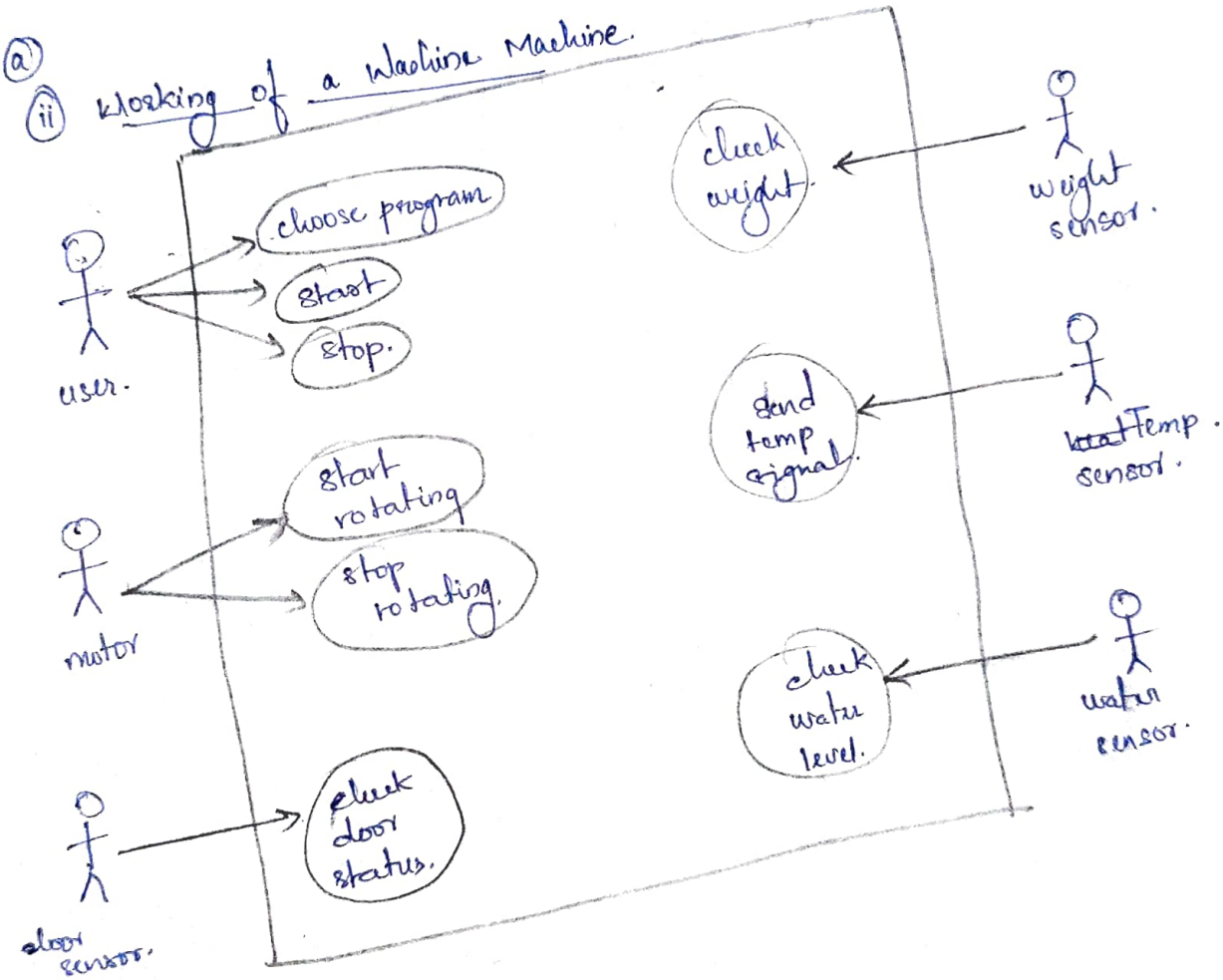
(2) (a)

(i) Smart cooking process in an oven.

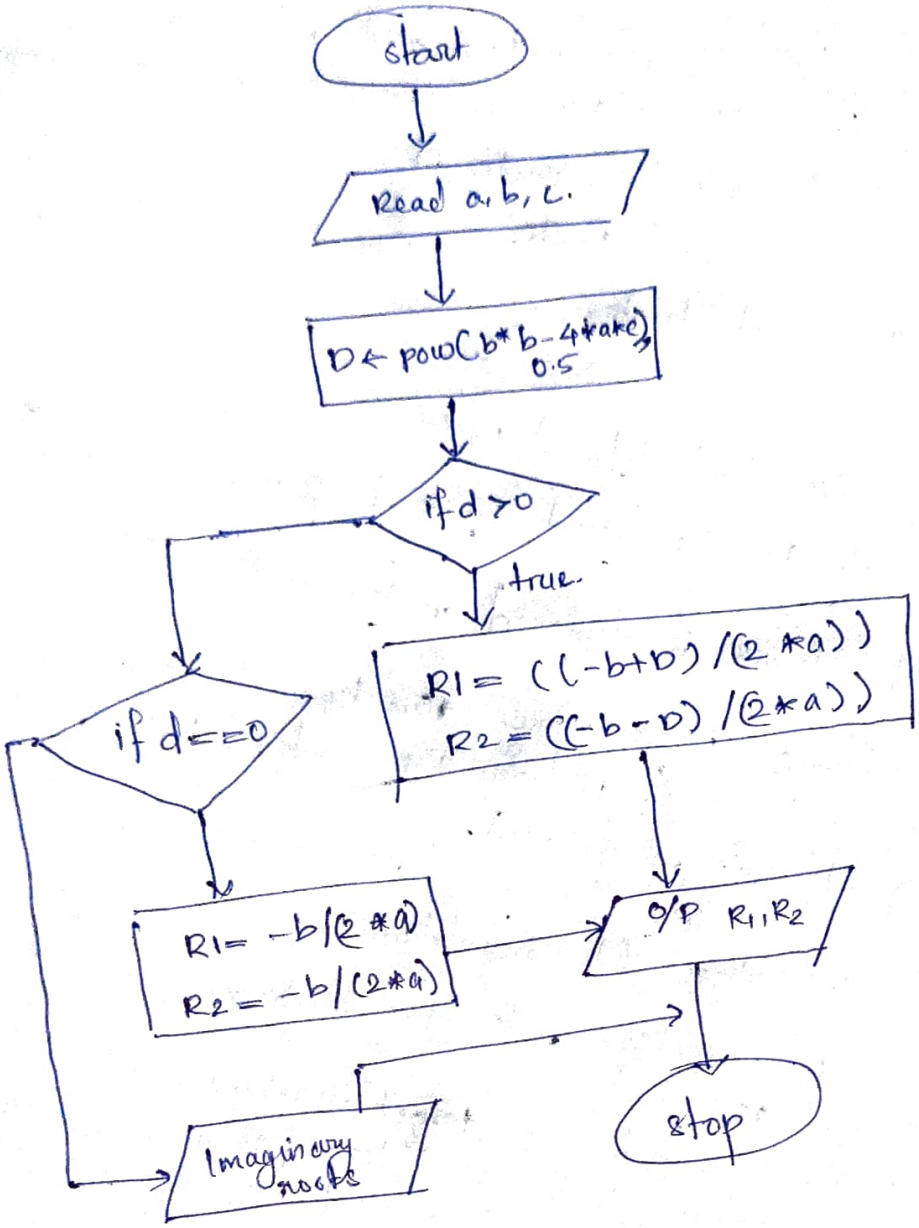


2 (a)

(ii) Working of a Washing Machine.



(2) (b)



(4) (b) Super Loop Based Approach

- The approach is applied for the applications that are not time critical and the response time is not so important.
- Similar to the conventional procedural programming where the code is executed task by task.
- Task listed at the top of the program code is executed first and task below the first task are executed after completing the first task.

(i) Configure the common parameter and perform initialization for various H/w components memory, registers etc.

(ii) start the first task & execute it.

(iii) Execute the second task.

(iv) Execute the next task.

(v) ....

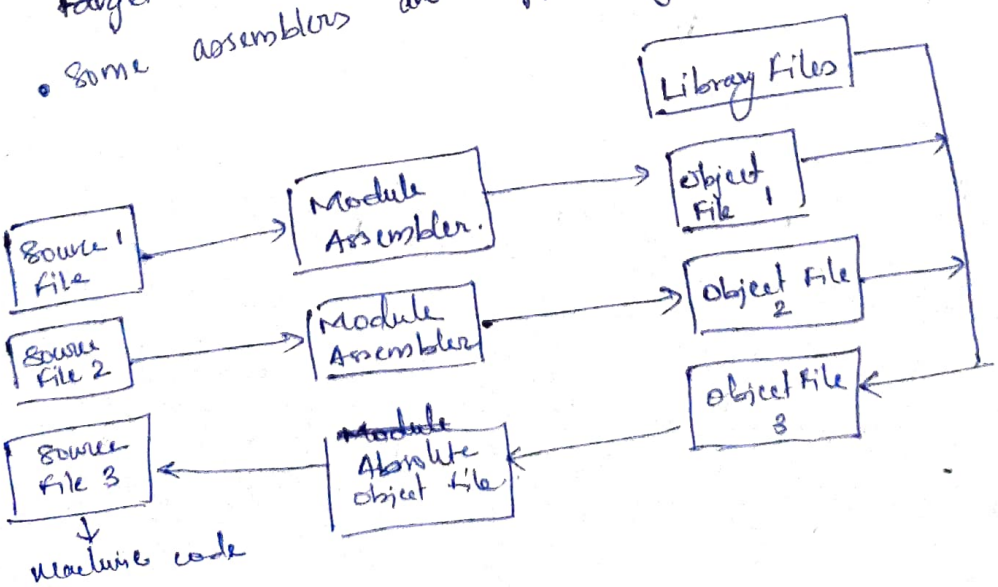
(vi) Execute the last defined task.

(vii) Jump back to the first task & follow the same flow.

(3) This type of design is deployed in low-cost embedded. this why electronic video game this type of firmware design approach. It will perform the task until or unless the stop operation is performed.

(4) (a) Soure file to object file translation

- Translation of assembly code to machine code is performed by assembler.
- the assemblers for different target machines are different and it is common that assemblers from multiple vendors are available in the market for the same target machines.
- Some assemblers are supplied by single vendor only.



(iv) Each source module is written in assembly and is .src or .asm file

(v) Each file can be assembled separately to examine the syntax errors and incorrect assembly instructions.

(vi) On assembling each .src file a corresponding object file is created with .obj extension.

(vii) Absolute address allocation is done at absolute object file creation stage

(viii) Each module can share variables and subroutine among them.

(ix) Exporting a variable from a module is done by declaring that variable as PUBLIC in source module.

(x) while assembling a module, on using variable with keyword EXTRN, assembler understand that these variables or function come from an external module and it proceeds assembling the entire module.

(6) (a) Types of files generated on Cross compiler.

the various files generated during cross compilation are:

1. List file
2. Preprocessor o/p file
3. Hex file (.hex)
4. Map file.
5. Object file (.obj).

(i) List file (.lst file)

- Generated at the time of cross compilation.
- contain information about cross compilation process like
  - Cross compiler details
  - Formatted source text.
  - Assembling code generated from the source file.
  - Symbol table.

(ii) Preprocessor o/p file

- contains preprocessor o/p for the preprocessor inst. used in the source file.
- The file is used for verifying the operation of macros & conditional preprocessor directives is a valid c file.
- file extension is cross compiler dependent.

(iii) Hex file.

- The hex file is an ASCII text file with lines of text that follow the Intel Hex file format.
- Intel Hex files are often used to transfer the program and data that would be stored in a ROM.

(iv) Map files

- object file created contains reallocated codes. i.e; their location in memory is not fixed.
- It is the responsibility of linker to link these object modules. these files are used to keep the information of linking & locating process. Map files use extensions .H, .HH depends on linker or loader.

(v) Object files

- It is the lowest level file format for any platform.
- Cross compiling each source module converts the various embedded instruction & other directives present in the module to an object (obj) file.
- OMF1 FORMAT 2 are the 2 object files supported by C51 compiler.

(6) (b) Code Reverse Engineering

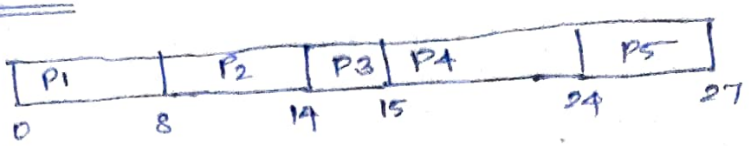
Using code reverse engineering we can identify the technology behind the efficient for machine.
Reverse engineering is the process of understanding the technology behind a product by extracting the information from the finished product.

Reverse engineering is performed by "hawkers" to reveal the technology behind the proprietary product.
Though most of the product employ code memory protection, if it may be possible to break the memory protection & read the code memory, it can easily be converted into assembly code using disassembler programs for the target machine.

SJC17CS035

(7)

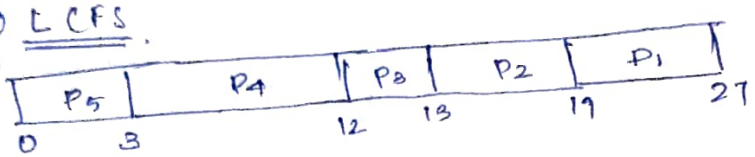
(a) (a) FCFS.



Avg. waiting time = $\frac{0+8+14+15+24}{5}$ = 12.2ms.

Avg. turn around time = $\frac{8+14+15+24+27}{5}$ = 17.6ms

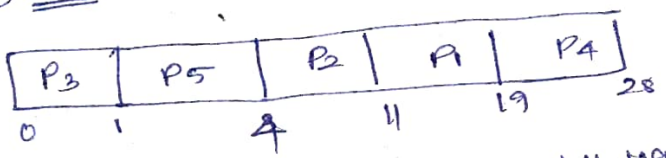
(7) (a) (b) LCFS.



avg. waiting time = $\frac{0+3+12+13+19}{5}$ = 9.4ms.

avg. turnaround time = $\frac{3+12+13+19+27}{5}$ = 14.8ms.

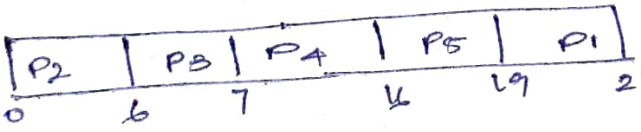
(7) (a) (c) SJF



Avg. waiting time = $\frac{0+7+4+11+19}{5}$ = 7ms.

Avg. turn around time = $\frac{1+4+11+19+28}{5}$ = 12.6ms.

(7) (a) (d) Priority based

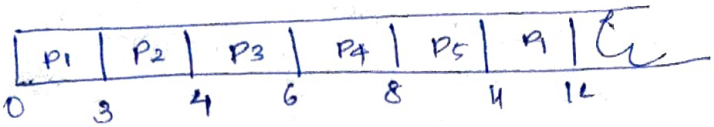


Avg. waiting time = $\frac{0+6+7+16+19}{5}$ = 9.6ms

Avg. turnaround time = $\frac{6+7+16+19+27}{5}$ = 15ms.

(7)

(a)(e) Round Robin (TS 3).



Avg. waiting time = $\frac{(12-8)+(4-6)+(6-1)+(8-9)+(11-3)}{5}$

= 2.8 ms.

Avg. T.A.T = $\frac{12+4+6+8+11}{5}$ = 8.2 ms.

(9) (a) Processor Trends

- System on clip.
- System in Package
- Multicore processor / clip level multiprocessor.
- Reconfigurable processor.

(i) System on clip

→ It makes a system on a clip.
→ Multiple functions can be performed on this clip.
→ Eg: iMX31.
→ Advantages
  → Save board space.
  → Leads to concept miniaturization.

(ii) System in Packages

→ In this sub systems are assembled into a single package.
→ It is characterized by one or more ICs of diff. functionalities which may include massive components assembled into a single package that functions as a system.
→ Adv.
  • Reduced time to market.
  • Reduced sizing.

(iii) Chip level multiprocessor

→ Incorporates multiple core processor on the same chip.
→ Works on the same clock frequency supplied to the chip.
→ eg: ARM Cortex-A provides a symmetric multi core.

(iv) Reconfigurable processors

→ It is a ~~multi~~ micro controller or processor with reconfigurable h/w features.
→ SoC has to be configured to the req. functionality through s/w support at the time of initialization.

Embedded OS trends

→ Most embedded os try to implement virtualization concept.
→ they adopt micro kernel architectures where only essential remaining appears as servies and placed in the user space.
→ There are os customized for the parts.
eg: MS Embedded os.

(9) (b) The decision of choosing an RTOS for an embedded design is very crucial. A lot of factors needs to be analyzed carefully before making a decision on the selection of RTOS.
These factors can be either functional or non functional.

Functional Requirements

(i) Processor support
(ii) Memory requirement.
(iii) Kernel & Interrupt Latency.
(iv) Inter process communication.
(v) Modularization support.

SIC17CS035.

(iv) Support for Networking & Communication.

(v) Development Language Support.

Non functional requirement

→ Custom developed

→ Cost

→ Development & debugging tools availability

→ Ease of use

→ After sales.