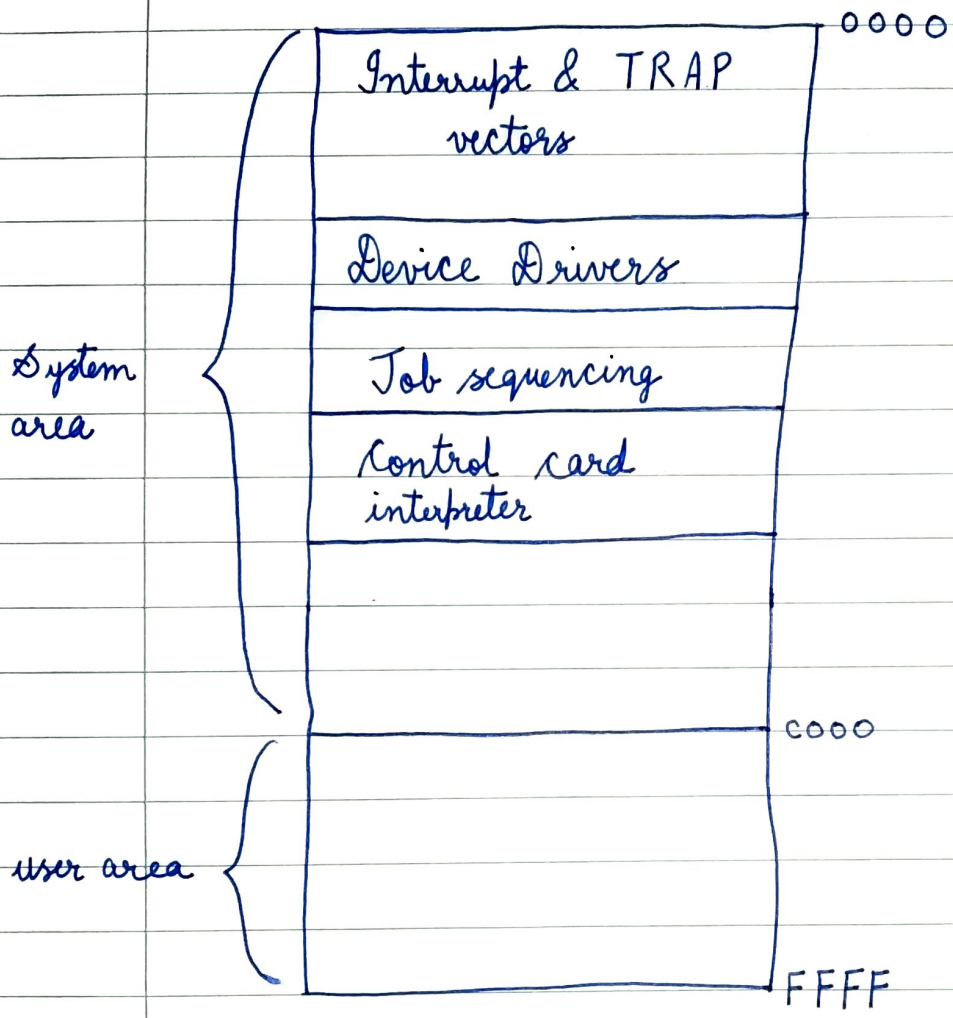
OS

Syllabus books
Galvin

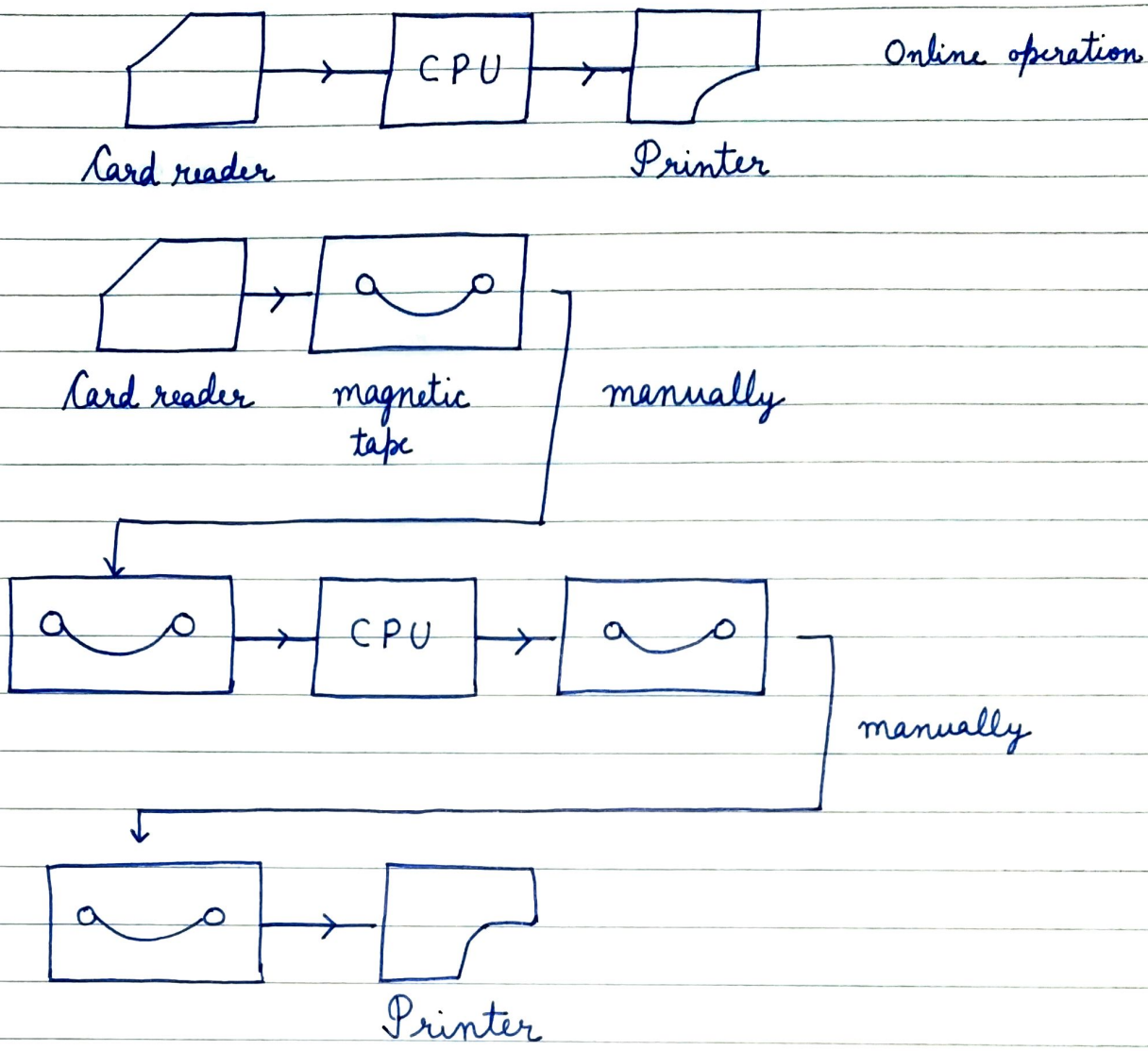
Note: OS of Dyna 85 was Resident Monitor



(linker-loader)

Source code → Object code → Executable code

Offline operations



CPU is faster than I/O devices. So, when read/ write operations are done, CPU is idle. To reduce this idle time, fast magnetic tapes were used. Now, the data is read/ written from slow device to fast magnetic tape, then magnetic tape is used as I/O device.

Buffering

Another solution of slowness of I/O devices is buffering.

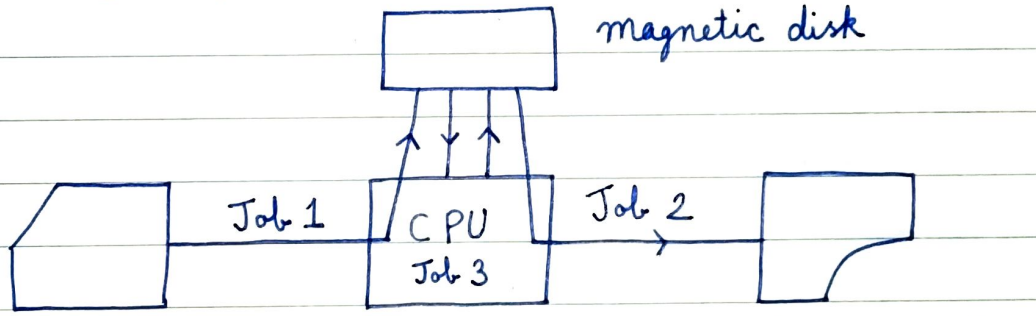
After data has been read and CPU has started operating on it, then the I/O device is instructed to begin the next input immediately. The CPU and I/O device both are busy then. Similar buffering for output devices also.

4/11/22

Spooling

The problem with magnetic tape is that card reader can't write on one end while CPU reads from other end

The entire magnetic tape is to be written before it is rewound and read



Spooling process

Simultaneous peripheral operation online (SPOOL)

Buffering overlaps I/O operations of one job with its computation whereas spooling overlaps I/O of one job with computation of another job.

Three simultaneous jobs are running during spooling

__/__/____

System calls

(a) Process control
- end, abort
- load, execute
- create process, terminate process
- get process attributes, set process attributes
- wait for time
- wait event, signal event

(b) File manipulation
- Create file, Delete file
- Open file, close file
- read, write, execute
- get file attributes, set file attributes

(c) Device manipulation
- Request device, Release device
- read, write, reposition
- get device attributes, set device attributes

(d) Information maintenance
- get time/date, set time/date
- get system data, set system data
- get process/file/device attributes
- ~~get~~ set process/file/device attributes

Systems program

(a) File manipulation
These program create, delete, copy, rename, print, list, manipulate file / directories

(b) Status information
- date, time, memory used / free, no. of users

(c) File modification
- Create / edit file, modify the contents

(d) Programming language support
- compilers, assemblers, interpreters

(e) Program loading and executing
- absolute loaders, relocatable loaders, linkage editor

(f) Application program
Compiler-compilers
Text formatters
plotting packages
~~locatable system~~ database system

Multiprogramming

The most important aspect of job scheduling is the ability to multiprogram. ~~Offer~~ Offline operations, buffering, spooling are for overlapped I/O and have limitations. A single user cannot keep CPU ~~busy~~ and I/O devices busy all the time, so multiprogramming concept came

__/__/____

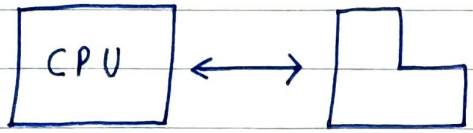
Concept:
The OS takes 1 job from the job pool and begins to execute it. Eventually, the job may have to wait for something, i.e., magnetic tape is to be mounted, I/O operation to be completed, etc. Now, CPU would sit idle in uni processing system. In a multiprogramming system, the OS will switch to next job and so on. Finally, the first job completed its I/O and ready to execute so CPU is always busy and never be idle.

10/11/22

Time Sharing
A time shared OS uses CPU scheduling and multi programming to provide each user a time shared operation.

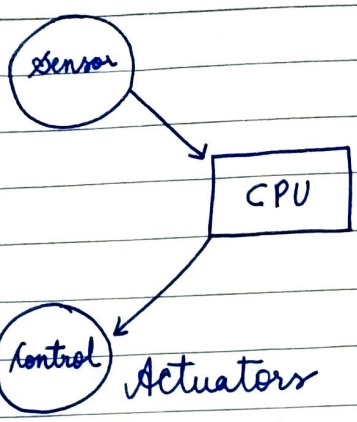
Batch processing

Card reader Printer

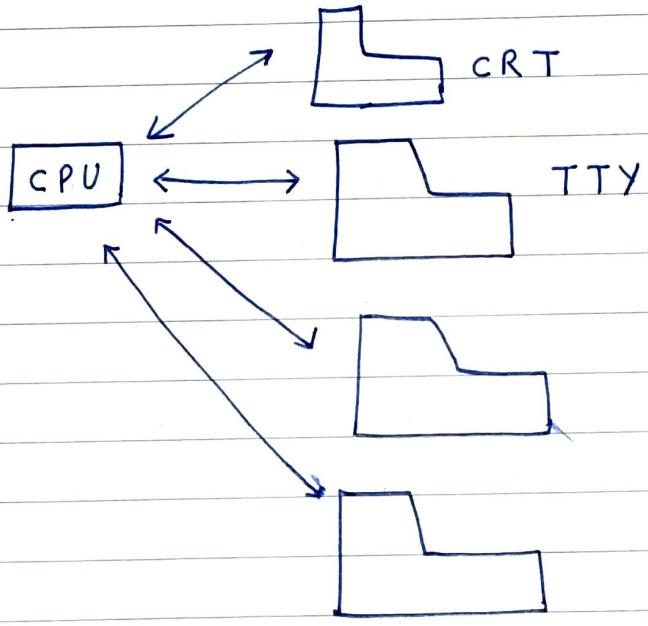


Terminal

single user interface



Real time system



Protection

1. Memory
2. CPU protection
3. I/O protection

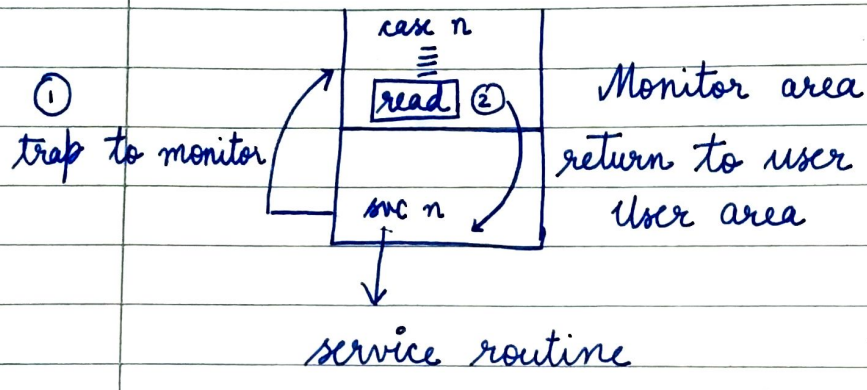
> As the concept like spooling came, one program might be executing while I/O activities might be occuring for other jobs

> Magnetic disks simultaneously held data for many jobs

> Multi programming holds several programs in memory at once
Sharing and error in 1 program might

affect other jobs. That's why protection is essential.

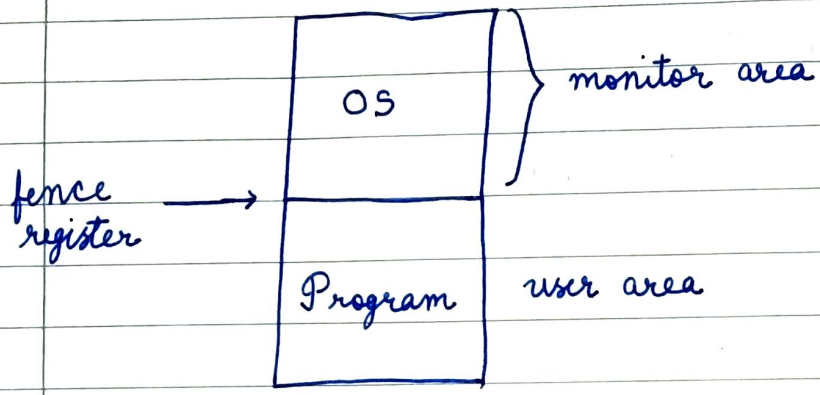
1. I/O protection



There are 2 modes of operation : user mode and monitor mode. All I/O operations can be performed in monitor mode. So when user ~~me~~ needs to perform I/O, it uses an instruction / ~~sca~~ system call. Whenever a system call is executed, control passes through a service routine in the monitor. The mode bit is set to monitor mode. Service routine will decide what user program is requesting, then it executes the request and return control to user.
This dual mode approach, allows OS to have complete control over computer system at all times. User program cannot directly do any I/O activity which might compromise the system's proper operation. They must request to monitor. The monitor first checks whether the request is reasonable and permissible, then only it's executed.
In addition, monitor can perform buffering, spooling, ~~b~~blocking to improve performance.

__/__/____

It can also map logical devices to physical devices to provide device independence.
So dual mode approach allows I/O protection and improved performance



Every address generated in the program is compared with fence register and attempt by a program to access system memory results in a TRAP to monitor (OS) which treats it as a fatal error. Setting fence register is a privilege instruction executed in monitor.

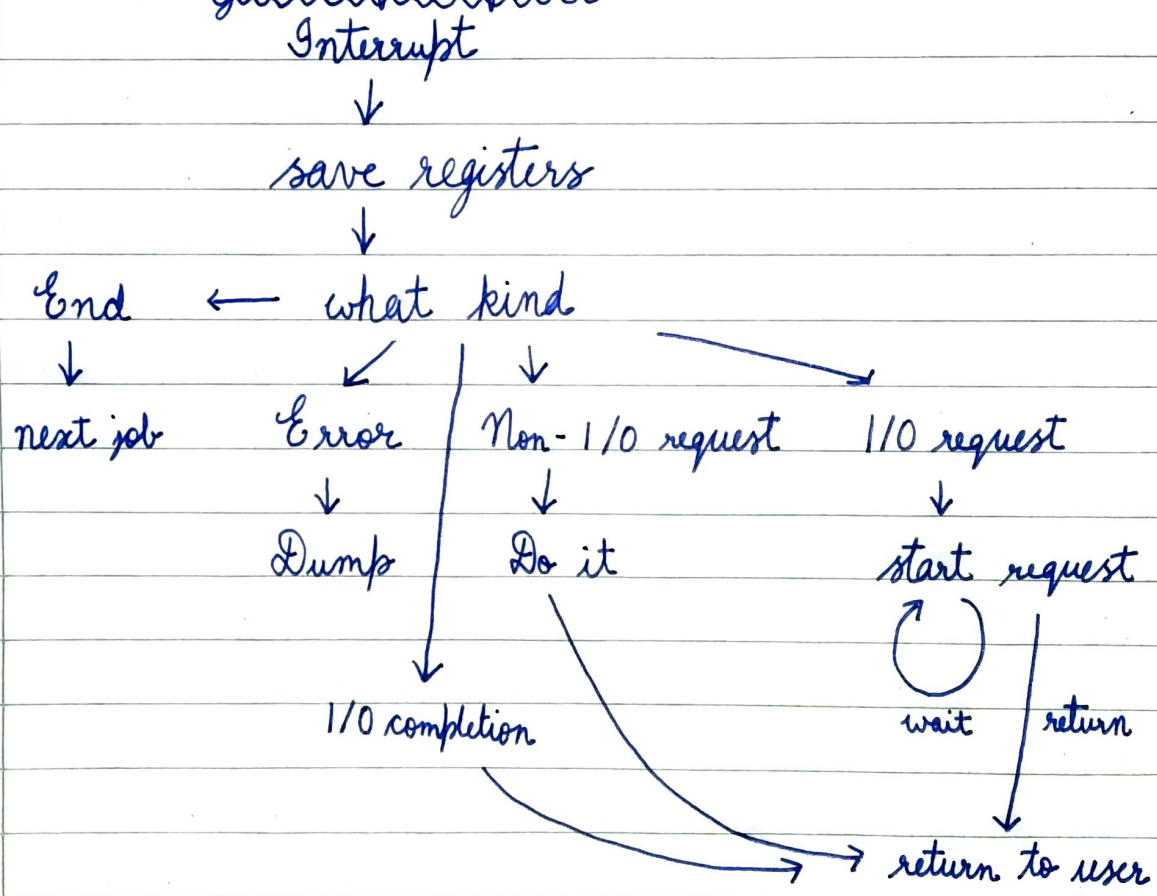
3. CPU protection

For this purpose we use timer, it will prevent user program from getting stuck in infinite loop and never returning control to the monitor.
When timer interrupts, the control transfers to monitor which may treat it as fatal error or decide to give the program more time.
To modify timer is a privilege instruction executed in monitor mode.

Operating system services

- program execution
- input / output operation
- File system manipulation
- Error detection
- Resource allocation
- Accounting
- Protection of all the hardware and software components

General flow of OS



Types of OS

1. Mainframe OS:

Mainframes have large I/O capacity. Normally there are three types of services:

- > batch processing — Eg: sale reports
- > transaction processing — Eg: cheque processing
- > time sharing — Eg: query

2. Server OS:

It runs on servers which are either PCs, workstations, mainframe systems. It serves multiple users over a network. It provides print service, file service, web service, etc.

3. Multi processor OS:

Multi processor is multiple CPUs working in a single system. Eg: Parallel computers, multi computers, multi processor, etc. that needs special features to communicate and connect.

4. PC OS:

For a single user, PC OS is used. Eg: It is used for word processing, spreadsheets, web surfing, etc.
Examples of PC OS: Windows 11, Ubuntu, etc

5. Realtime OS:

Here time is the key parameter. Eg: In industrial process control system, the real time compiter has to collect data about the production process and use it to control the machine in factory acc. to deadline

They are of two types:
> Hard realtime systems
> Soft realtime systems

6. Embedded OS:
A palm top computer, PDA is a small computer that fits in a small pocket and performs small number of functions such as electronic address book and memo pads.
Embedded systems run on computers that control devices that are not generally thought of as computers such as TV sets, microwave ovens, telephone sets. These often have some characteristics of real time systems but also have size, memory and power restrictions that make them special.
Eg: Palm OS, Window CE

7. Smart Card OS:
Smallest OS run on smart cards (credit cards containing CPU) They have very severe processing power and memory constraints.
Some of them can only have a single function such as electronic payments.
Some smart cards are java oriented. Java applets are downloaded to the card and are interpreted by the JVM interpreter.

OS

(i) Memory
(ii) Processor
(iii) I/O devices
(iv) File / information management

Memory management

1. Keeping track & status of each memory word
2. Allocation policy - to whom
   how much
   when
   where
3. what allocation technique is to be used:
   It tells about specific location selected and info updated.
4. Deallocation policy and technique - when release or OS will acquire the allocated memory

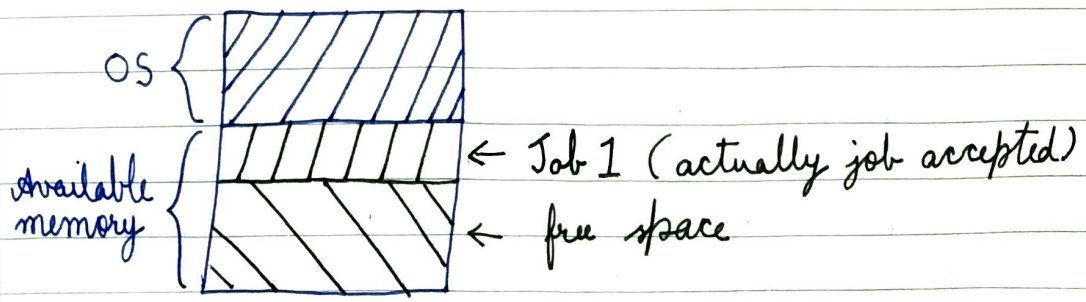
8.

Memory management policies ~~do~~ in two different ways:

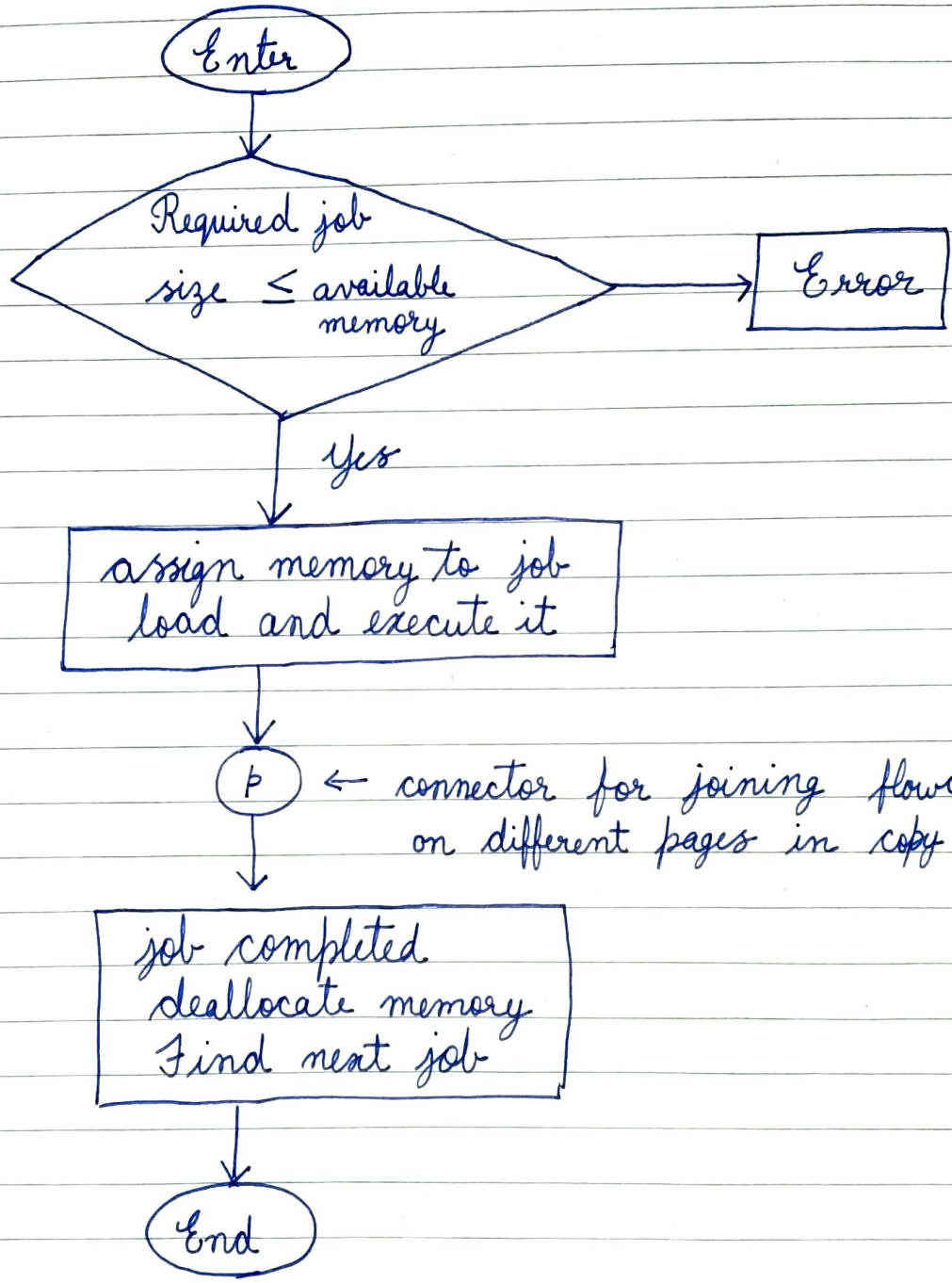
(i) greater utilization of memory
(ii) more flexibility to user

Memory management schemes

a) Single contiguous allocation



- It is the simplest memory management scheme, no special hardware required
- Normally used for batch OS



Disadvantages

1. Poor utilization of memory
2. Poor utilization of processor (waiting I/O)
3. User's job being limited to the size of available memory

b) **Partitioned Allocation**

- one of the simplest approach for supporting multiprogramming.
- Main memory is divided into partitions
- Hardware support:

A memory protection mechanism is desirable to prevent one job to modify another job or OS. Solution: bound registers / fence registers

Bound register

Problems: They must be changed everytime a new job is multiprogrammed.

It is difficult to extend this protection to I/O activities

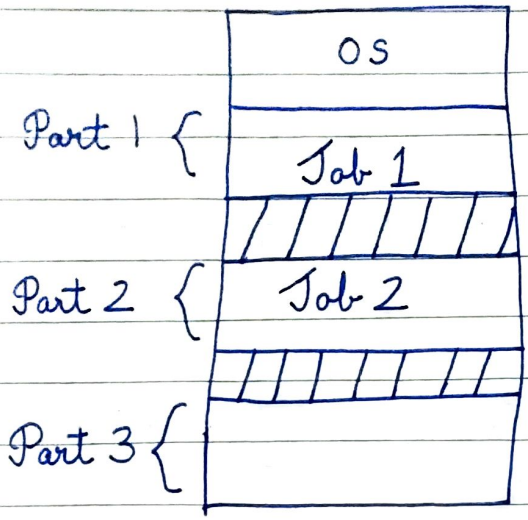
~~Software algorithm~~

Software ~~of~~ algorithm [static partioning specification
dynamic partioning specification

(i) Static partioning specification

- Memory is divided into partitions prior to processing of job

Partition No. (Protection key)	Size	Location	Status
1	8K	312K	in use
2	32K	320K	in use
3	120K	352K	not in use
4	520K	504K	in use



Each job must specify the maximum amount of memory needed. A partition of sufficient size is then found and memory allocated.

Pros:

- This technique is appropriate when size and frequency of job are well known.

Eg: memory requests of job coming to execution:
1K, 9K, 33K, 121K ---

Partition No. (Partition Key)	Partition size	Job size	Wasted size
1	8K	1K	7K
2	32K	9K	23K
3	120K	33K	87K
4	520K	121K	399K
	680K	164K	516K

$$\% \text{ Wastage} = \frac{516}{680} \times 100 = 75.8\%$$

(ii) Dynamic Partition Specification

Partition No ↓ PN

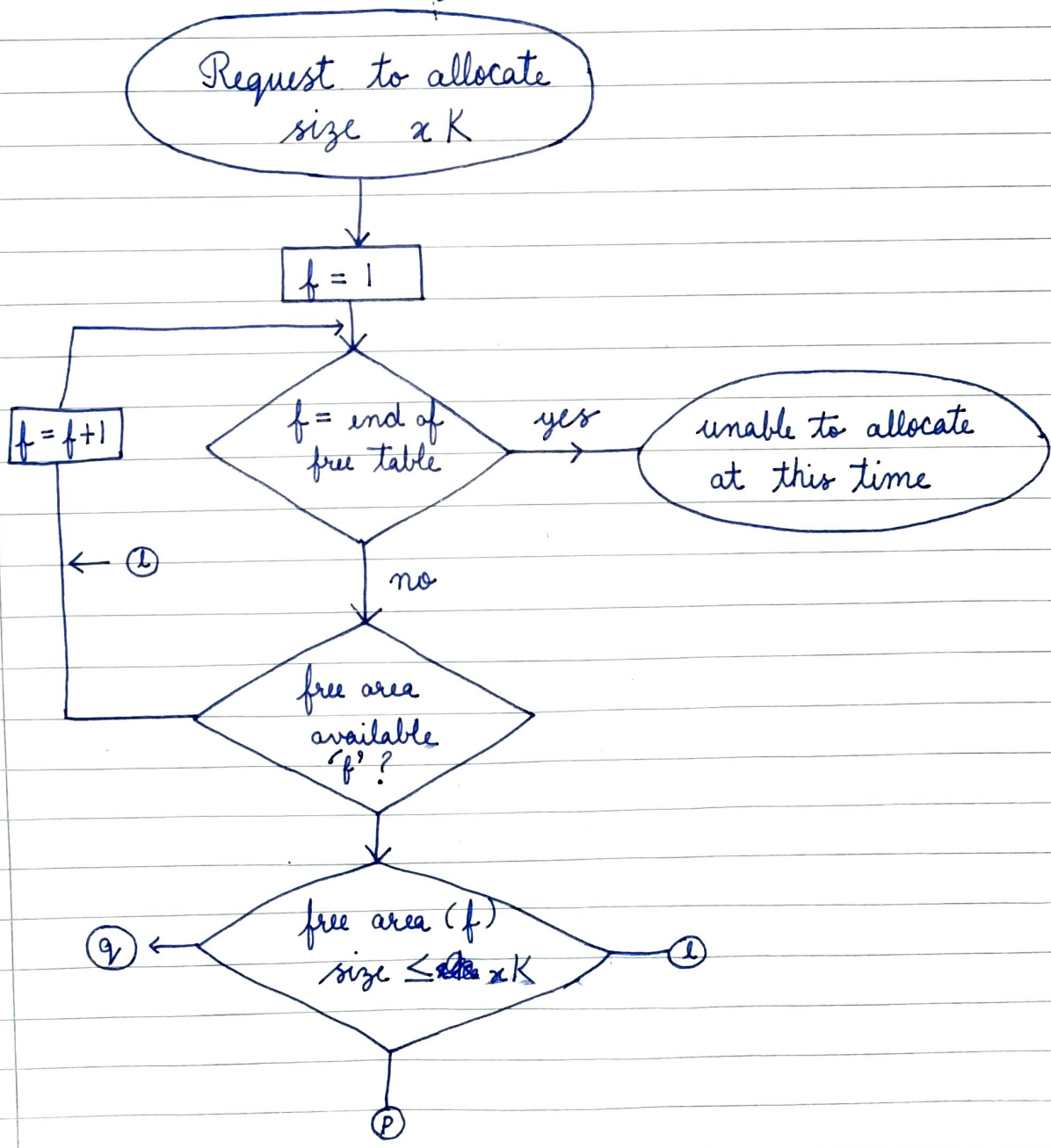
Allocation table

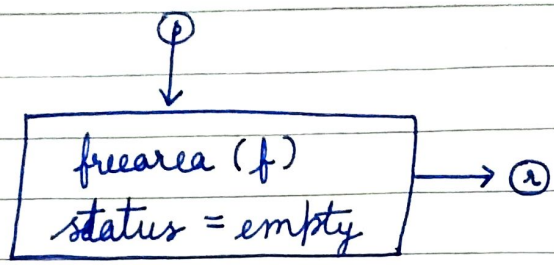
Size	Location	Status
8K	312K	Allocated
32K	320K	Allocated
–	–	Empty
120K	384K	Allocated
–	–	Empty

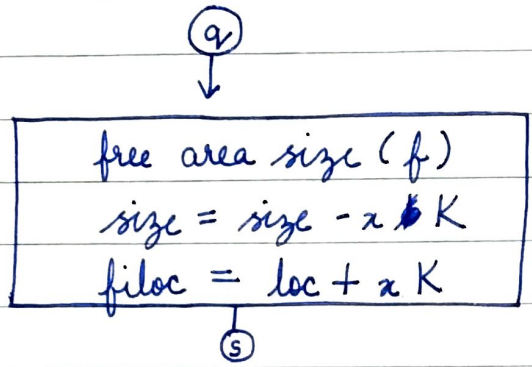
Free area ↓ F.A.

Free Area table

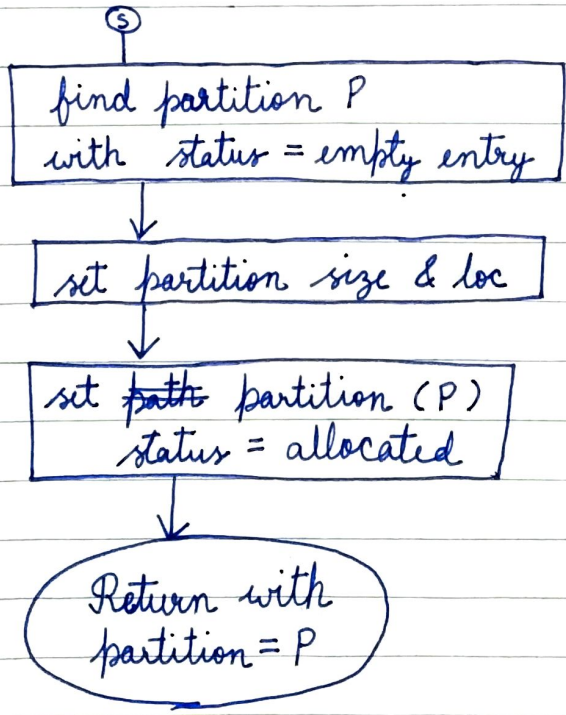
F.A.	Size	Location	Status
1	32K	352K	Available
2	520K	504K	Available
3	–	–	Empty
4	–	–	Empty
5	–	–	Empty



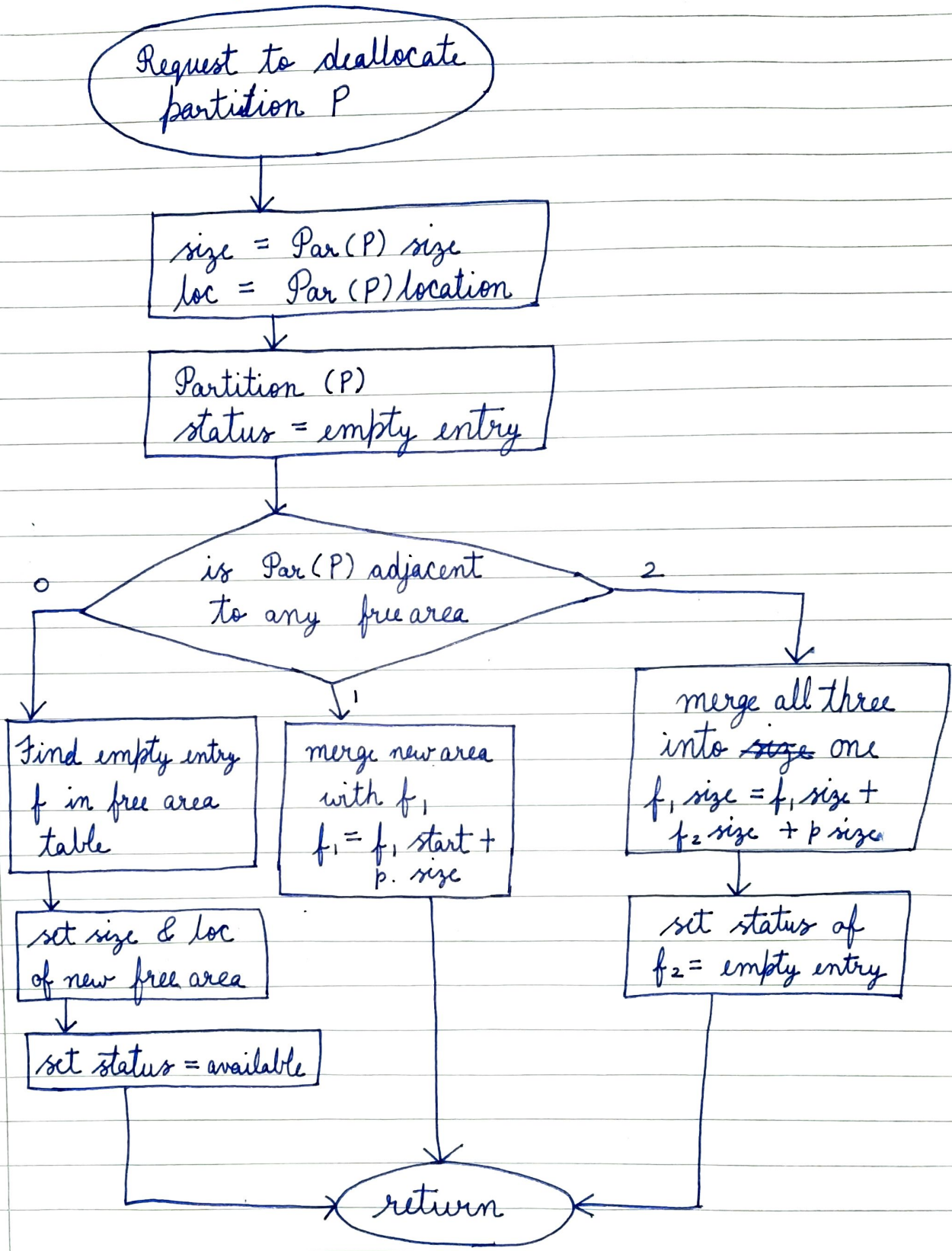

p
freearea (f)
status = empty
r


q
free area size (f)
size = size - x K
filoc = loc + x K
s


new job
size


s
find partition P
with status = empty entry
set partition size & loc
set ~~path~~ partition (P)
status = allocated
Return with
partition = P

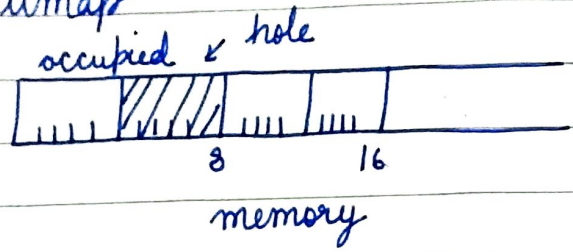
Deallocation algorithm

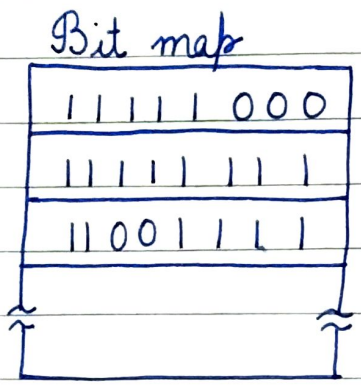


It can be represented in two ways:

1. Bit map
2. Linked list

1. Bitmap





Memory is divided into allocation units.
Corresponding to each allocation unit there is a bit in the bitmap
This bit is zero if free and one if occupied

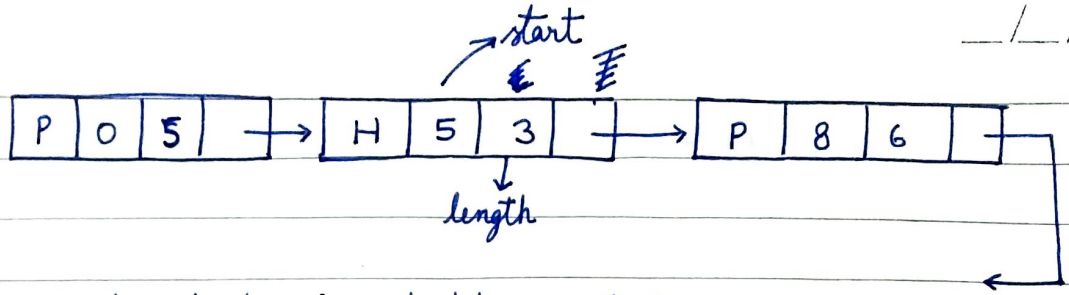
Bitmap provides simple way to keep track of memory words

Problems:
When mem manager has to allocate memory to k-unit process, it has to search the bitmap for k consecutive zero bits in the process but searching is a slow operation

2. Linked List
Another way of keeping track of memory is linked list of allocated free memory segments.
A segment is either a process or hole (edge between two process)

Note: best fit uses ascending order of list of hole (sorted by size)

__/__/__



Segment list is kept sorted by address
Sorted lists are beneficial. When a process terminates or is swapped out, then updation is easy.

Eg:

(i) 

If this process terminates, it will be shown as

Similarly

(ii) 

(iii) 

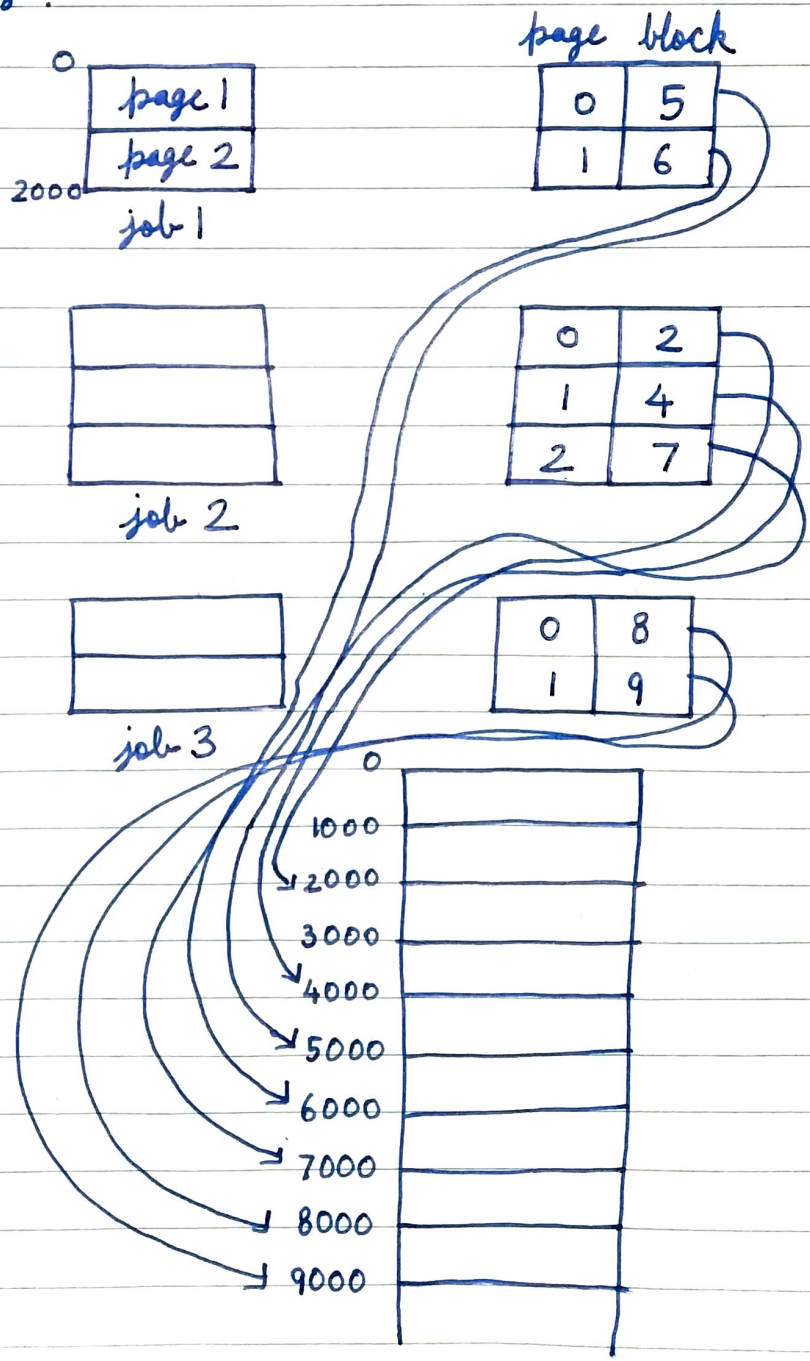
(iv) 

When the list is sorted by address, several algorithms can be used to allocate memory

1. First fit: from the first entry (search)
2. Next fit: from current pointer
3. Best fit: from 1 to last to find the most suitable hole
4. Worst fit: biggest hole search
5. Quick fit: it maintains separate list for holes

Paged memory management

- Each job's address space is divided into equal pieces called pages.
- Likewise, physical memory is divided into pieces of same size called blocks.
- Now any page can be mapped into any block. The pages remain logically contiguous but corresponding blocks are not necessarily contiguous.



The four functions of memory management are performed as follows:

1. Keeping track of status
   (i) Page map table
   (ii) Memory ~~map table~~ block table

2. Determining who gets memory (decided by memory scheduler)

3. Allocation: All pages of job must be loaded into assigned blocks and appropriate entries are mapped in the PMT and MBT.

4. Deallocation: When the job is done, blocks must be returned setting the status in MBT as free.

Hardware support

- A hardware mechanism is needed to map from each ~~address~~ instruction effective address to physical memory allocation.
- If address space of a job is small, we may use high speed register for each page.
- If 100 K bytes are available in memory & block size is 4 K bytes.
- If address space is used, using page map table is better.

- Relationship among pages, blocks, PMTAR, PMTs

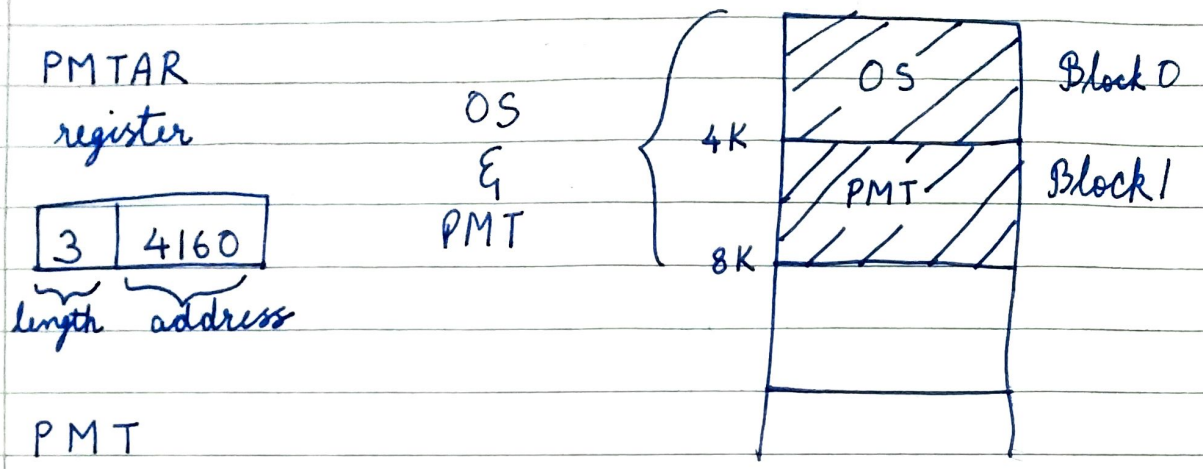
PMTAR register

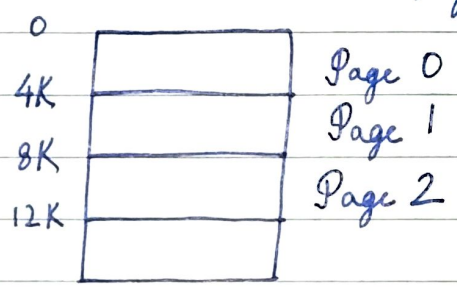
3	4160

length address

PMT

0	2
1	4
2	7





job 2 address space

There are 3 basic tables to be managed by OS:

1. Job Table (JT)
2. Memory ~~Blo~~ Block Table (MBT)
3. Page Map Table (PMT)

Each job has separate entry in job table (JT) indicating the location and length of its PMT and status information.

MBT indicates status of each memory block whether it is allocated or available

Job No.	Size	location of PMT	Status
1	8K	3600	Allocated
2	12K	4160	Allocated
3	4K	3820	Allocated
4	-	-	Empty

	Page No	Block No.
(3600)	0	5
(3601)	1	6

job 1 PMT

Page No.	Block No.
0 (4160)	2
1 (4161)	4
2 (4162)	7

job 2 PMT

Page No.	Block No.
0 (3820)	8

job 3 PMT

MBT

Block No.	Status
0	OS
1	OS
2	job 2
3	available
4	job 2
5	job 1
6	job 1
7	job 2
8	job 3
9	available

- Overlapping of address space
  If address space has a page no. of another job, then protection must be provided by the OS so that program cannot modify it. Either make it read-only or use lock-key mechanism so that a single key can be provided to a single job.

Advantages

1. It eliminates fragmentation (external)
2. It has higher degree of multiprogramming, better processor and memory utilisation.
3. The compaction overhead required for partition scheme is also eliminated.
4.

Disadvantages

1. Page map addressing (hardware) increases the cost of the system and reduces the processing speed.
2. Memory is used to store PMT and other tables, so less memory is available to the user.
3. Internal fragmentation is still a problem.
4. If required no. of blocks are not available, then these blocks are not allocated and space is wasted even though it is free
5. Job's address space is limited to the size of physical memory.

__/__/____

Demand Paging

PMT hardware may have a status bit that a page is assigned a block ~~or~~ or not.
If a page is referenced which is not present in the physical memory then it is a page fault.
Now that needed page must be loaded from secondary memory to main memory and status is set to yes.
Initially only page 0 is loaded in main memory, other pages are loaded with page fault occurence but when all blocks are allocated then where the new page will be loaded then an existing page is replaced.
Which page will be replaced will be decided by page replacement algorithm.

Thrashing: If frequent page faults occur and substantial ~~wastage~~ computer time is wasted in just replacing the pages, this phenomenon is known as thrashing.

H/W support

H/W of paged memory management plus the following:

- → status bit in PMT
- → interrupt action to transfer control to OS if page fault
- → record of individual page usage to determine which is to be replaced by OS

S/W support

→ A job's pages are stored is secondary storage so each page is having a file address and a file map table is maintained.

→

MBT

0	OS
1	OS
2	job 2, page 0
3	available
4	job 2, page 1

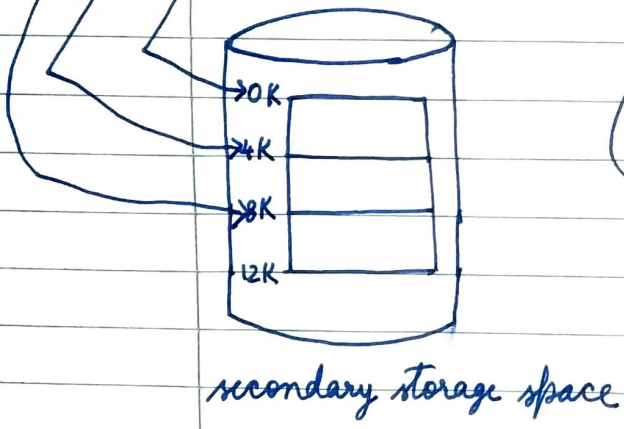
PMTAR

File Map Table (job 2)

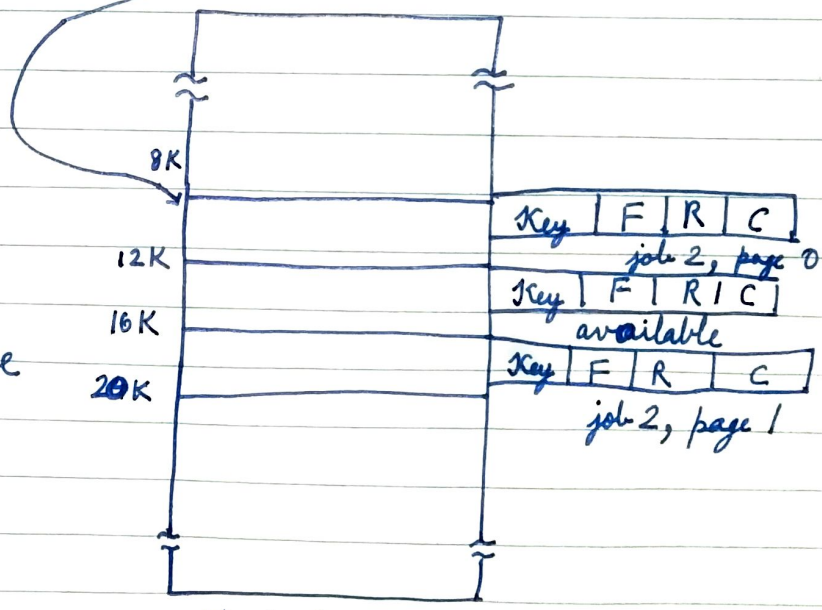
Page	
0	
1	
2	

PMT

Page	Block	I
0	2	0
1	4	0
2	–	2



secondary storage space



physical memory

Key - protection key for block
F - free
R - referenced
C - changed

__/__/____

Q1. How to find a particular page in main memory

A1 It is answered by the status of the page invalid bit in PMT entry

Q2. Is there a free block?

A2 It is answered by examining the MBT of available block

Q3. Was page changed?

A3. It is answered by examining changed bit. (whenever any write to that page then changed bit is set). If not changed then copy of page on secondary memory is same, so no need to write this page back.