MEMORY ORGANISATION

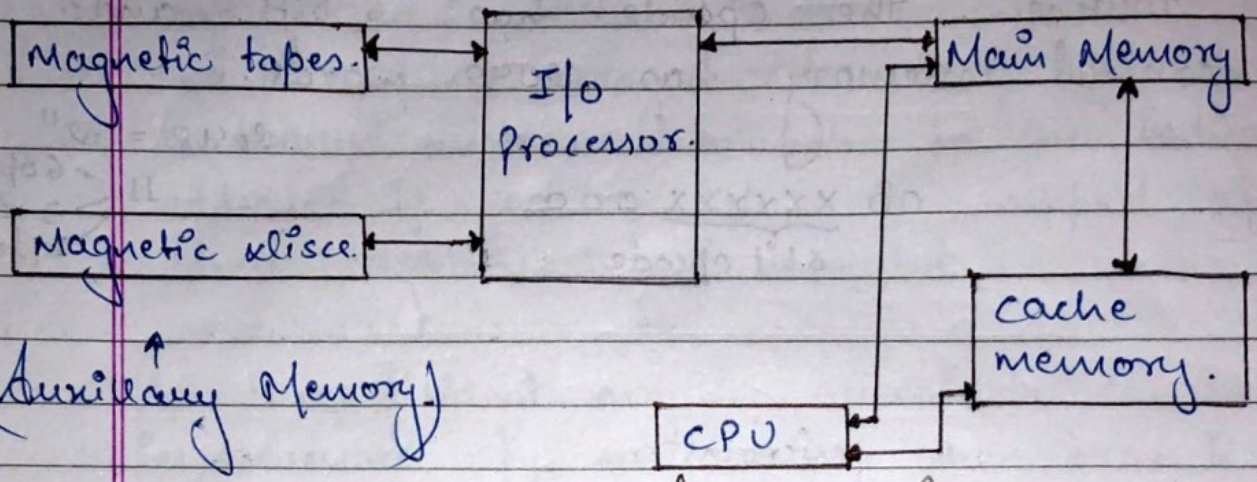
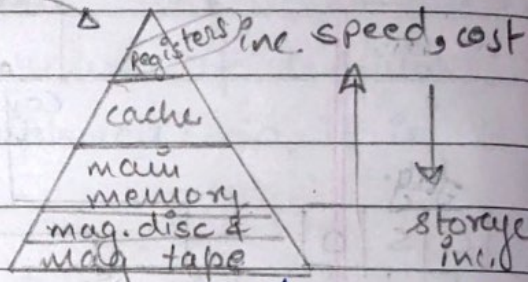


Fig: Memory hierarchy



→ The main goal of memory hierarchy is to obtain highest possible average access speed by minimizing the total cost of entire memory system.

volatile memory
↓
static & dynamic
RAM.
static RAM
→ flip flop
Dynamic RAM
→ capacitor
ROM → non volatile & bootstrap

Main Memory.

- It is relatively large & fast memory used to ~~storag~~ store programs & data during the computer operation.

The principle technology used for main memory is based on semiconductor integrated circuits. Integrated RAM chips are available in two possible modes- static & Dynamic.

→ The static RAM consists of internal flipflops that stores the binary information. The stored information remains valid as long as power is applied to the unit.

→ The Dynamic Ram stores the binary information

(1024x8)

512 bytes RAM → 128x8 $= \frac{512 \times 8}{128 \times 8} = 4$ RAM chip

512 bytes ROM → 512x8 $= 1$ ROM chip.

Date:
Page:

in the form of electric charges that are applied to the capacitors. The stored charge on capacitors tends to discharge with time & the capacitors must be periodically recharged by refreshing the dynamic memory.

volatile

- RAM is used for storing bulk of programs & data that are subject to change.
  ROM is used for storing program that are permanently resident in computer.
- ROM portion of main memory is needed for storing an initial program called a bootstrap loader.
  The bootstrap loader is a program whose function is to start the computer software operating when power is turned ON.

non-volatile

RAM chips & ROM chips.

(7 bit address bus) (8 bit data bus)

1024x8 memory constructed with 128x8 RAM chips & 512x8 ROM chips.

[space taken by 128 RAM = space taken by 512 ROM]

we can take 128 ROM chip also but to prove above equality we're designing with 512 ROM.

17/10/18

RAM chip

A RAM chip has one or more control inputs that select the chip only when needed.

RAM chip select only when
$\overline{CS1} = 1$
$\overline{CS2} = 0$

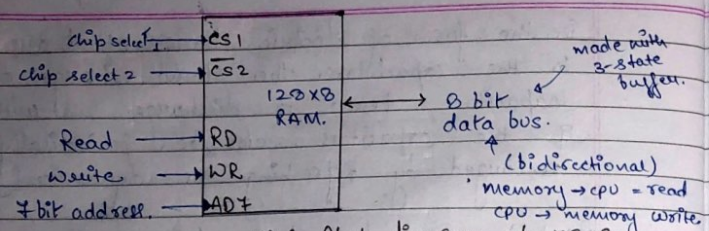


Fig: Block diagram of 128x8 RAM chip.

CS1	$\overline{CS2}$	RD	WR	Memory function	State of data bus
0	0	x	x	Inhibit	high impedence state
0	1	x	x	inhibit	high imped. state.
1	0	0	0	inhibit	high impedence state.
1	0	0	1	write	Input data to RAM.
1	0	1	x	Read.	Output data from RAM
1	1	x	x	inhibit	high impedence state

Functional Table

ROM chip.

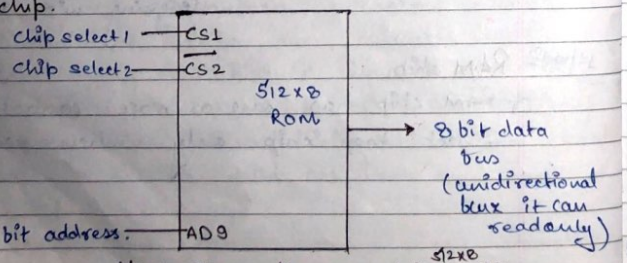


Fig: Block diagram of ROM chip.

* Internal cells of ROM chip take less space compared to RAM chip.

CS1	$\overline{CS2}$		Memory func.
0	0	high impedence	Inhibit
0	1	high impedence	Inhibit
1	0	Read	Read
1	1	high impedence	Inhibit

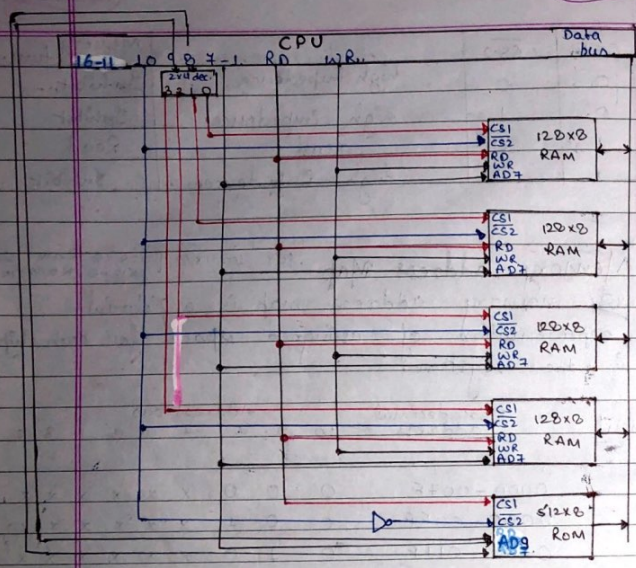
Memory address Map.

for 1024x8 [128x8 RAM size/chip, 512x8 ROM size/chip]

→ A memory address map is a pictorial representation of assigned space for each chip in the system.

component	Hexadecimal address	Address bus (16-11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1)
RAM 1	0000 - 007F	0 0 0 x x x x x x x x
RAM 2	0080 - 00FF	0 0 1 x x x x x x x x
RAM 3	0100 - 017F	0 1 0 x x x x x x x x
RAM 4	0180 - 01FF	0 1 1 x x x x x x x x
ROM	0200 - 03FF	0 1 x x x x x x x x x

Memory connection to CPU



Ques. a) How many 128x8 RAM chips are needed to provide a memory capacity of 2048 bytes.
b) How many lines of address bus must be used to access 2048 bytes of memory. How many of these lines will be common to all chips.
c) How many lines should be decoded for chip select & specify the size of decoder.

a) $2048/128 = 16$

b) 1 to 7 + 4 select for 16 RAM = 7+4 = 11
(1 to 7 : common)

c) 4 → 4x16 decoder.

Ques. A computer uses RAM chips of 1024x1 capacity

a) How many chips are needed & how should there address lines be connected to provide a memory capacity of 1024 bytes. (1024x8)
(1 to 7 common; 3 decoder; 3x8)

b) How many chips are needed to provide a memory capacity of 16K bytes. Explain in words how the chips are to be connected to the address bus.

a) Ram chip = $\frac{1024 \times 8}{1024 \times 1} = 8$ RAM chip.

8 chips are needed with add. line connected in parallel.

b) $\frac{16K \times 8}{1024 \times 1} = \frac{2^4 \times 2^{10} \times 8}{2^{10}} = 128$ RAM chip.

$16K = 2^{14}$

14 13 12 11 | 10-1

decoder.

16 Ram chip.

Ram 1 … Ram 16 : 8 bit data bus.

But 16, each 8 parallely connected.

128 chips are needed use 14 address lines, 10 lines specify chip address, 4 lines are decoded into 16 chip select i/p.

Ques. A computer employs RAM chips of 256×8 & ROM chips of 1024×8. The comp. sys needs 2K bytes of RAM, 4K bytes of ROM and 4 interface units, each with 4 registers. A memory mapped I/O configuration is used. The 2 highest order bits of address bus are assigned 00 for RAM, 01 for ROM & 10 for interface um registers. a) How many RAM & ROM chips are needed.
b) Draw a memory address map for system.
c) Give address range in hexadecimal for RAM, ROM & interface.

a) RAM chips = $\frac{2 \times 1024 \times 8}{256 \times 8} = 8$ (256×8)

ROM chip = $\frac{4 \times 1024 \times 8}{1024 \times 8} = 4$ (1024×8)

interface reg. = 4×4 = 16

Components	Address	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
RAM	0000-07FF	0 0 0 0 0 xxx x x x x x x x
ROM	4000-4FFF	0 1 0 0 x x x x x x x x x x
Interface	8000-800F	1 0 0 0 0 0 0 0 0 0 0 0 x x x x

RAM 1 0000-00FF ROM1 → 4000-43FF
RAM 2 0100-01FF ROM2 → 4400-47FF
RAM 3 0200-02FF ROM 3 - 4800-4BFF
ROM4 - 4C00-4FFF
RAM 8 0700-07FF

Ques. Draw memory address map for 4096 byte of RAM & 4096 byte of ROM.
Size of 1 RAM chip 128×8, ROM chip 512×8

No. of RAM = $\frac{4096 \times 8}{128 \times 8} = 32$

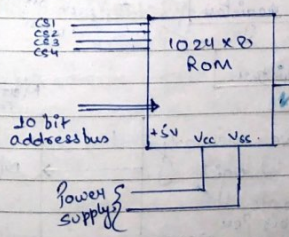
ROM chip = $\frac{4096 \times 8}{512 \times 8} = 8$

Components	Address	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
RAM	0000-0FFF	0 0 0 0 5×32 decoder x x x x x x x
ROM	1000-1FFF	0 0 1 0 3×8 decoder x x x x x x x x

Ques. A ROM chip of 1024×8 bits has 4 select inputs and operates from a 5V. power supply. How many pins are needed for the IC package. Draw a block diagram & label all ips and outputs terminal in ROM.

8 + 4 + 2 + 10 = 16 + 8 = 24

data bus, chip sel., +5V, V_{cc}, V_{ss}



Temporal locality - means current data or instr that is being fetched may be needed soon. so we should store that data in Cache memory to avoid again searching in main memory for same data and saving time.

Associative Memory (content-addressable memory) CAM

→ The time required to find an item stored in memory can be reduced if stored data can be identified for access by the content of data itself rather than by an address.

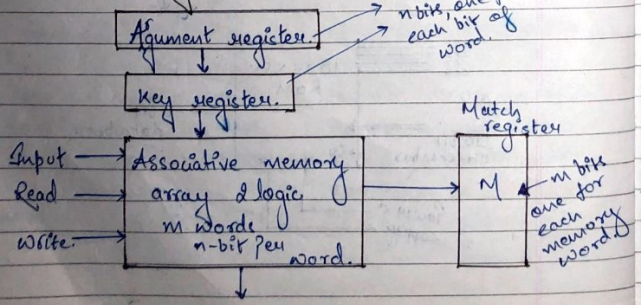
→ A memory unit accessed by content is called an associative memory or content addressable memory.

→ When a word is written in associative memory no address is given. the memory is capable of finding an empty unused location to store the word. when a word is to read from associative memory, the content of the word, or part of the word is specified.

→ An associative memory is more expensive than a RAM because each cell must have storage capability as well as logic circuits for matching its content with an external argument.

25/10/19

Hardware Organisation



spatial - instr or data near to the current memory location that is being fetched, may be needed soon (in near future)

```
sum = 0;
for (i=0; i< arrlength; i++)
    sum += arr[i];
return sum;
```

Ex.

A : 101 111100
K : 111 000000

sum is being used again & again - Temporal Locality
value of arr is being accessed in order arr[0], arr[1], ... - to show spatial locality.

word 1. 100 111100 no match
word 2. 101 000001 match.

Each word in memory is compared in parallel with the content of argument register. the words that match the bits of argument register set a corresponding bit in the match register. After the matching process, those bits in match register that have been set indicate that these corresponding words have been matched.

The key register provides a mask for choosing a particular field for key in argument word. The entire argument is compared with each memory word. if key register contains all 1's. Otherwise only those bits in argument that have 1's in corresponding position of key register are compared.

CACHE MEMORY

If the active portion of program & data are placed in fast small memory, the average memory access time can be reduced. Such a fast small memory is known as cache memory.

Basic operation of cache

When the CPU needs to access memory the cache is examined, if the word is

found in cache, it is read from the cache memory. If the word addressed by CPU is not found in cache, main memory is accessed to read the word.
A block of word is then transferred from main memory to cache memory.

Hit ratio = $\frac{\text{no. of hits}}{\text{total CPU references. (hit+miss)}}$

Performation of cache memory.

hit ratio more, more performation.

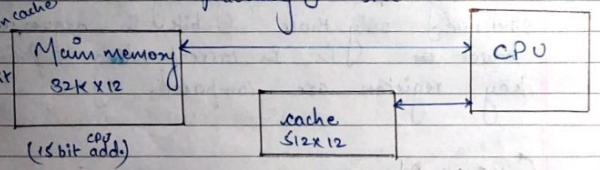
26/10/18

The transformation of data from main memory to cache memory is known as Mapping Process
There are 3 types of mapping -

1) Associative mapping.
2) Direct mapping
3) set associative mapping

Ex- Out of 32K word 512 are used frequently so in cache
32K → word
each word = 12bit

Consider main & cache memory of following size -



1) Associative mapping
The fastest & most flexible cache organisation uses an associative memory.
The associative memory stores both the address and content of the memory word.

1st search in cache if not found search in main memory provide to CPU as well as store in cache

03777 | 7453

CPU address (15 bits) (Provided by CPU to search)

Argument register

If cache is full the we apply replacement method (FIFO, Round Robin)

Address (octal)	Data (octal)
01000	3450
02777	6710
22315	1234
03777	7453.

Associative Mapping cache

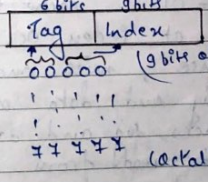
2) Direct mapping.
Associative memory are expensive compared to RAM because of added logic associated with each cell.

(15 bit) CPU address

Tag (15-9=6 bits) | Index (addl. bit of cache) 9-bit

000
111
777.

* In general if we have,
2^k words in cache.
2^n word in main memory
n bit memory address. has -
index = k bits
Tag = n-k bits.



Date: 26/10/18
Page:

CPU 00000 (1st) tag | index

Write in cache

02777 (2nd) tag | index

Address	Data
00000	1220
00777	2340
01000	3450
01777	4560
02000	5670
02777	6710

index address	Tag	Data
000	00	~~1220~~
	01	3450
777	02	6710

cache

CPU tries to check if data is available in cache.
it again gives address 00000
CPU breaks into 2 parts

00 (tag) 000 (index) → search in cache.

and matches with tag of cache memory & its own tag.

if both same then hit.

If CPU gives 01000 (read from cache else find in main)
tag index

CPU tag = 01 cache tag = 00
at index 000

Hence, miss condition, so finds it in main memory, tag and data corresponding to 01000 so replaces tag and data of cache & overwrites that read from main.

(In such practice hit ratio falls it is a disadv.)

Date: 29/10/18
Page:

Disadvantage of direct mapping is that the hit ratio can drop if two or more words whose addresses have same index but different tags are accessed repeatedly.

If 1 block = 8 words.

$\frac{512}{8} = 64$ blocks.

15 bit CPU add.

(6) Tag | Index (9)

block (6) | word (3) (Octal 3 bit)

no. of blocks = 2^6

Each block has 2^3 words.

* More block size, more hit ratio improvement

(Octal) index	Tag	Data
000 (word)	01	4370
block 0		
007	01	5610
010	02	—
block 1		
017	02	—
		—
770	03	—
block 63		
777	03	—

Tag within each block should be same.

3) Set Associative mapping.

Set associative mapping is an improvement over the direct mapping where each word of cache can store 2 or more words of memory under the same index address.

Each data word is stored together with its tag and the number of tag data items in one word of cache is said to form a set

02000 ctu address

Index	Tag	data	Tag	data
000	02 (not found)	3450	02 (store here)	5670
777	--	--	--	--

Two way-set associative.

index = 512

Tag = 6 + data = 4x3 = 12 = 18

Two word tag = 18 x 2.

word = tag + data (6) (12) (18)

Size of cache memory = 512 x 36

Ques. A digital computer has a memory unit of 64K x 16 and a cache memory of 1K words. The cache uses direct mapping with a block size of 4 words.

a) How many bits are there in tag, index, block and word field of address format.

b) How many bit are there in each words of cache. and how are they divided into functions. include a valid bit.

c) How many blocks can cache accomodate.

↳ when we power on, cache memory initialize, and it is not empty it has non-valid data. when in first miss cond. we bring data from main we set valid bit = 1 and it is equal to 1 till we don't apply any replacement.

30/10/18

Ques. A 2 way set associative cache memory uses blocks of 4 words. The cache can accomodate a total of 2048 words from main memory. The main memory size is 128K x 32. Formulate all the information required to construct the cache memory

b) what is size of cache memory.

a) $2^{16} \times 16$

$1K = 2^{10}$

Tag: 16 - 10 = 6 | index: 10

index → Block (8 bits) | word (2 bits)

8 bits in block field.

c) $2^8 = 256$ (Verify. $256 \times 4 = 1024$ in cache)

b) word = tag + data

V	Tag	data
1	6	16

1 + 6 + 16 = 23 bits.

a) $\frac{2048}{2} = 1024 = 2^{10}$ = 10 bit (index) address.

17 bit: Tag (7 bit) | index (10 bit) → block (8) | word (2 bit)

Tag | index

7 bit | 10 bit

block (8) bits in block | word (2 bit)

no. of block = $2^8 = 256$

b) 1024×78 data = 32, Tag = 7] = 39 x 2 = 78

1024 x 64

Date: 30/10/18
Page:

Writing into cache

When a CPU finds a word in cache during read operation the main memory is not involved in the transfer. If the operation is a write there are two ways the system can proceed.-

1) Write through method.
The simplest and most commonly used procedure is to update main memory with every memory write operation, with cache memory being updated in parallel. If it contains the word at specified address. This is called write through method.
This method has the advantage that main memory always contains the same data as the cache.

2) Write back method.
In this method, only the cache location is updated during a write operation. The location is then marked by flag. so that, later when the word is removed from cache it is copied into main memory.

VIRTUAL MEMORY

~~In a memory hierarchy~~
Virtual memory is a concept used in some large computer system that permit the user to construct programs as a large memory space is available equal to the totality of auxiliary memory.

add. space = set of virtual add.
mem. space = set of physical add.

Date:
Page:

Each address that is referenced by the CPU goes through an address mapping from virtual address to a physical address in main memory.
Virtual memory system provides a mechanism for translating program generated addresses into correct main memory locations.
The translation or mapping is handled automatically by hardware by means of a mapping table.

Address space & Memory space
An address used by programmers is virtual address and set of such addresses is address space.
An address in main memory is called location or physical address and set of such location is called memory space.

31/10/18

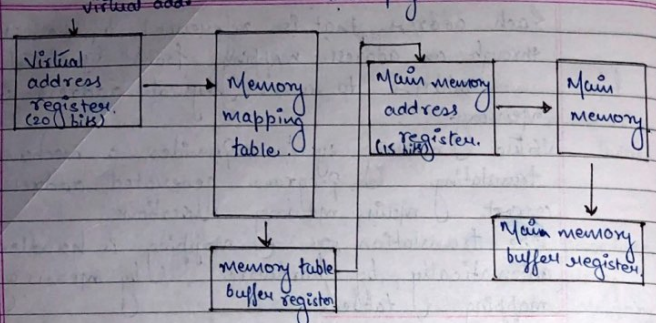
The mapping table may be stored in separate memory or in main memory.
In first case, an additional memory unit is required as well as one extra memory access time. In second case, the table takes space from main memory and two accesses to memory are required. with the program running at half speed.

for ex: N = 1024 K bits.

virtual add. = 2^{20} = 20 bits

Main memory = 32K bit

~~main~~ physical = 15 bits



Address mapping using Pages

* memory space → blocks
* address space → pages.

1 Page size = 1 block size = 1K words.

no. of pages = $\frac{1024K}{1K} = 1024$

no. of blocks = $\frac{32K}{1K} = 32$

Page → map → block.

~~ex~~ virtual add. < Line number, Page number

→ The table implementation of address mapping is simplified if the information in the address space and memory space are each divided into groups of fixed size.

→ The physical memory is broken down into groups of equal size called blocks and the term Page refers to groups of address space of same size.

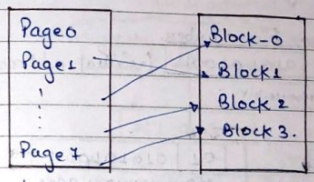
If address space = 8K, virtual add = 13 bit

memory space = 4K, Physical add = 12 bit

1 Page size = 1 block size = 1K word.

no. of pages = $\frac{8K}{1K} = 8$

no. of block = $\frac{4K}{1K} = 4$



Max. 4 page can reside in main memory.

virtual address (13 bit)

Page Number (3 bits) | Line number (10 bits) address within a page.

If 1 page has 1024 words

Line address = 10 bit.

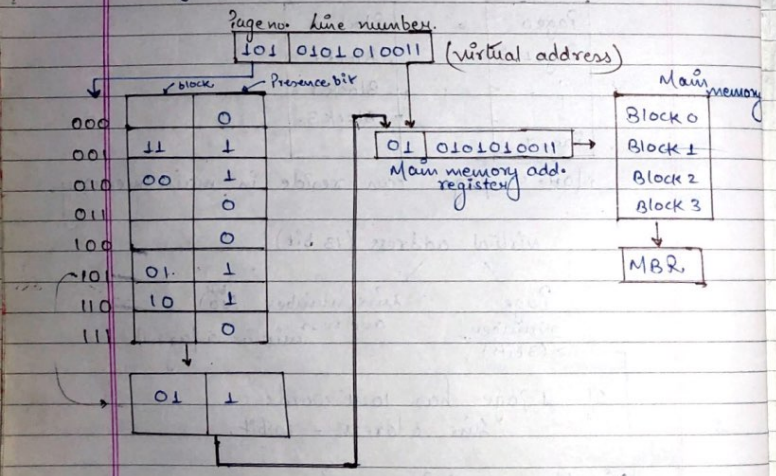
→ implies no. of pages = 2^3

when Page → block, corresponding line number doesn't change.

Block number = 2 bit (since we've 4 blocks)

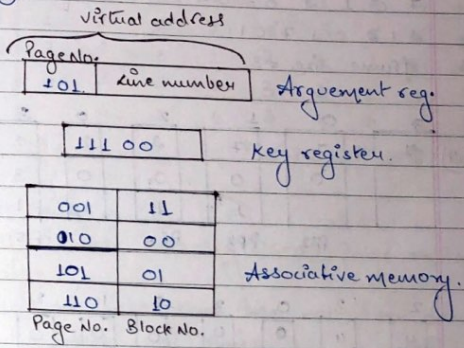
Date: 31/10/18
Page:

The mapping from address space to memory space is facilitated. If each virtual address is considered to be represented by 2 numbers, A page number address and a line within the page.
In a computer with 2^p words per page., p bits are used to specify line address and the remaining high order bits of virtual address specify the page number.



1/11/2018

Associative Memory Page table. (no storage wastage)
A more efficient way to organise the page table is to construct it with a number of words equal to the number of blocks in main memory. In this way, size of memory is reduced and each location is fully utilized. This method can be implemented by means of an associative memory with each word in memory containing a page number together with its corresponding block number.
The page field in each word is compared with page number in virtual address, if a match occurs the word is read from memory and its corresponding block number is extracted.



Page replacement algorithms.
When a program start execution one or more pages are transferred into main memory and the page table is said to indicate their position. The program is executed from main memory until it attends attempts to reference a page that is still in auxiliary memory. This condition is called page fault. When a page fault occurs in virtual memory system it signifies that the page referenced by the CPU is not in main memory. A new page is then transferred from auxiliary memory to main memory. If main memory is full, it is necessary to

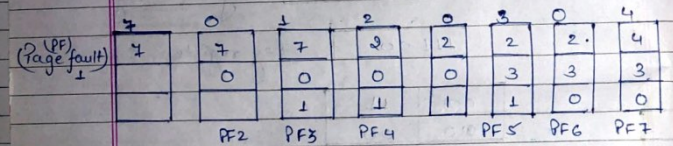
Remove a page from memory block to make room for new page. The policy for choosing pages to remove is determined from the replacement algorithms.

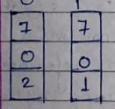
1) FIFO Page replacement.

Given –
frame size = Max. Page in main memory.
Reference string.

For ex reference string =
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

frame size = 3





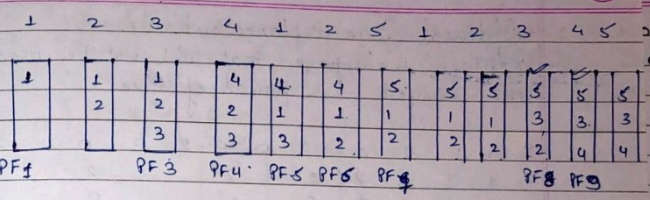
Total Page fault = 15

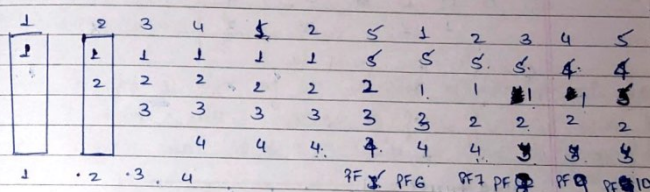
Belady's anomaly

Reference string.
1 2 3 4 1 2 5 1 2 3 4 5

(i) frame size = 3
Page fault = 9.

(ii) frame size = 4
Page fault = 10





2) LRU (Least recently used) Page replacement algo.

Replace the page that has not been used for longest period of time.

Ex. String –
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

frame size = 3

Page fault = 12.

Scanned with CamScanner

3) Optimal Page replacement.
Replace the page that will not be used for the longest period of time.

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2
7	7	7	2	2	2		2		2			2		
	0	0	0	0	0		4		0			0		
		1	1		3		3		3			3		

0 1 7 0 1
7
0
1

Page fault = 9.

Ques. An address space is specified by 24 bits and the corresponding memory space by 16 bits.
a) how many words are there in address space.
b) how many words are there in memory space.
c) If a page consists of 2k words then how many pages & blocks are there in the system.

a) 2^{24} words
b) 2^{16} words
c) no. of pages = $\frac{2^{24}}{2^{11}} = 2^{13}$

no. of blocks = $\frac{2^{16}}{2^{11}} = 2^{5}$

magnetic disc
tape

Ques. A virtual memory has a page size of 1K words. There are 8 pages and 4 blocks. The associative memory page table contains the following entries.

Page	block
0	3
1	1
4	2
6	0

Make a list of all virtual addresses in decimal that will cause a page fault if used by CPU.
Page no. 2,3,5,7 will cause a page fault.

Pg No.	Add.	Address that will cause page fault
2	2K	2048 - 3071
3	3K	3072 - 4095
5	5K	5120 - 6143
7	7K	7168 - 8191

0 → 0-1023
1 → 1024 → 2047
2 → 2048 → 3071
3 → 3072 - 4095
4 → 4096 - 5119
5 → 5120 - 6143
6 → 6144 - 7167
7 → 7168 - 8191

Ques:- Consider a system with 2 level cache. The access time of L1 cache, L2 cache & main memory are 1ns, 10ns, 500ns. The hit ratio of L1 and L2 cache are 0.8 and 0.9 repectively. What is the average access time.
t_1 = 1ns, t_2 = 10ns, t_m = 500ns; L1 cache $h_1 = 0.8$; L2 cache $h_2 = 0.9$

Hierarichal access
$T_{avg} = h_1 t_1 + (1-h_1)h_2(t_1+t_2) + (1-h_1)(1-h_2)(t_1+t_2+t_m)$
$= 0.8 + 0.2 \times 0.9 \times 11 + 0.2 \times 0.1 \times 511$
$= 13ns$

Simultaneous access
$T_{avg} = h_1 t_1 + (1-h_1)h_2(t_1+t_2) + (1-h_1)(1-h_2)t_m$
$= 0.8 + 0.2 \times 0.9 \times 10 + 0.2 \times 0.1 \times 500 = 12.6$

what is the average access time of the system ignoring the search within the cache?

=

Consider the design of a level hierarchy with the following specifications for the memory characteristic

Memory level	Access time	capacity	Cost / kB
cache	$t_c = 25ns$	$S_c = 512\ kB$	$C_c = \$125$
Main memory	$t_m = -$	$S_m = 32\ MB$	$C_m = \$0.2$
Disc array	$t_d = 4ms$	$S_d = -$	$C_d = \$0.0002$

The goal is to achieve an average memory access time $t_a = 10.04\ \mu sec$ with a cache hit ratio $H_c = 0.98$ and $H_m = 0.9$ in main memory. Also the total of the memory hierarchy is upperbounded by $15,000. Find out the unknown quantities.

$$t_a = H_c T_c + (1-H_c) H_m t_m + (1-H_c)(1-H_m) t_d \leq 10.04 \times 10^{-6}$$
$$= 0.98 \times 25 \times 10^{-9} + (1-0.98)\,0.9\,t_m + (1-0.98)(1-0.9) \times 4 \times 10^{-3}$$
$$t_m = 1.11 \times 10^{-4}\ sec$$
$$= 111\ \mu sec.$$

$$512 \times 1.25 + 32 \times 1024 \times 0.2 + S_d \times 0.0002 \leq 15,000$$
$$S_d = 39032000\ kB = 37.22\ GB$$

Ques. A virtual memory system has an address space of 8k words. Memory space of 4k words and page & block size of 1k words. The following page reference changes occur during a given time interval.

4 2 0 1 2 6 1 4 0 1 0 2 3 5 7 find the page fault if the replacement algo. is

i) FIFO ii) LRU iii) optimal.

i) FIFO

Page size = 4

4	4	4	4	4	6	6	6	6	6	6
	2	2	2	2	2	2	4	4	4	4
		0	0	0	0	0	0	0	0	0
(1)	(2)	(3)	1	1	1	1	1	1	1	1
					(5)		(6)			

6	6	5	5
4	4	4	7
2	2	2	2
1	3	3	3
(7)	(8)	(9)	(10)

10 Page faults.

ii) LRU

4	4	4	4	4	4	4	4	4	4	4	4	4
	2	2	2	2	2	2	2	2	2	2	2	2
		0	0	0	6	6	6	0	0	0	0	0
			1	1	1	1	1	1	1	1	1	3

4	4
2	2
5	5
3	3

optimal.

4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
	2	2	2	2	6	6	6	6	6	6	2	2	2	2
		0	0	0	0	0	0	0	0	0	0	3	5	5
			1	1	1	1	1	1	1	1	1	1	1	7

9 Page fault.