COMPUTER ARITHMETIC

The four basic arithmetic operations in computers are: Addition, Subtraction, Multiplications, and Division.

- Arithmetic processor is part of processor unit and it executes arithmetic operations.
- Arithmetic instructions specify data type binary or decimal , fixed point or floating point
- Fixed point numbers represents integers while floating point numbers represent fractional numbers.
- Negative numbers may be in signed magnitude or signed complement representation.

o Fixed point binary signed magnitude

o Fixed point binary 2's complement

o Floating point binary

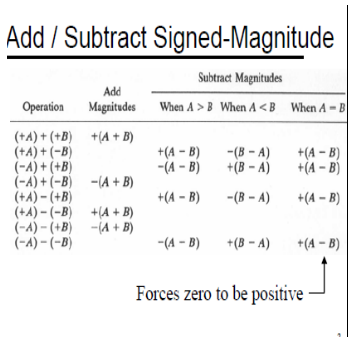
o Floating point BCD

Addition and Subtraction

1. Signed magnitude

- Is familiar as used in every day arithmetic calculations

- We designate magnitudes of two numbers as A and B. when those sign numbers are added we have eight different conditions depending on the sign bits and operations performed. Next table shows those conditions and other columns shows operations performed.



Add / Subtract Signed-Magnitude

Operation	Add Magnitudes	Subtract Magnitudes		
		When $A > B$	When $A < B$	When $A = B$
$(+A) + (+B)$	$+(A + B)$			
$(+A) + (-B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(-A) + (+B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$
$(-A) + (-B)$	$-(A + B)$			
$(+A) - (+B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(+A) - (-B)$	$+(A + B)$			
$(-A) - (+B)$	$-(A + B)$			
$(-A) - (-B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$

Forces zero to be positive

Addition (Subtraction) Algorithm will be as

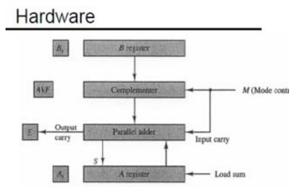
o When A and B have identical (different) signs, add the two magnitudes and attaché sign of A to result

o When signs of A and B are different (same), subtract smaller number from larger number. Choose sign of result as sign of A if A>B, or complement of sign of A if A<B.

o If two magnitudes are equal, subtract B from A and make sign of result positive.

Hardware Implementation

- The two numbers are stored in registers A and B and their signs in flip-flops As and Bs. Result is transferred to A register with its sign
- Next figure the hardware of signed magnitude addition-subtraction. Consists of registers A and B, sign bits As and Bs
- Complementer can pass value of B or invert of B according to value of M (implemented with EX-OR).
- AVF holds overflow bit when A and B are added.
- The addition of A and B will be through parallel adder.



 When M=0, B is transferred to adder with A and output in A = A + B. while M=1 complement of B plus carry=1 is transferred to adder with A and output A = A –

The algorithm is shown in next figure
o The two signs As and Bs are compared by EX-OR them. If result is 0 then As = Bs and if result is 1 the As ≠ Bs.

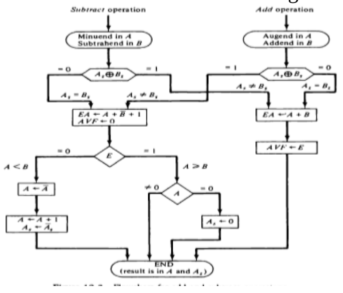
o For add operations if have same sign bits the magnitude must be added. For subtract operations different sign bits means magnitudes be added as well.

o E bit is carry bit after addition and moves to AVE overflow bit only at this state.

o If sign bits are different in add operations or the same in subtract operations the two magnitudes will be subtracted A – B. No overflow can occur here.

o After subtract if E=1 this means A>B and if E=0 then A<B. then here it is necessary to get 2's complement of A (by invert A then add 1) and sign of A is inverted only in this case.

o Final result will be in A and its sign in As.


Subtract operation
Add operation
Minuend in A Subtrahend in B
Augend in A Addend in B
$A_s \oplus B_s$
$A_s = B_s$
$A_s \neq B_s$
$EA \leftarrow A + \overline{B} + 1$ $AVF \leftarrow 0$
$EA \leftarrow A + B$
$AVF \leftarrow E$
E
A < B
A > B
$A \leftarrow \overline{A}$
$A \leftarrow A + 1$ $A_s \leftarrow \overline{A}_s$
$A_s \leftarrow 0$
END (result is in A and A_s)

Signed 2's complement

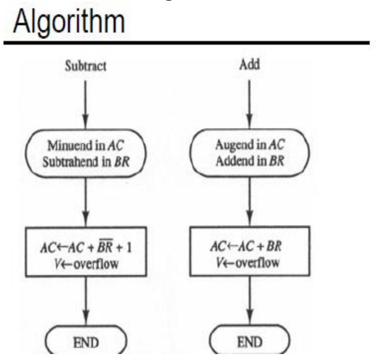
 The left most bit in 2's complement represented binary number is the sign bit. If 0 the number is positive and if 1 then number is negative. If sign bit is 1 the entire number is represented in 2's complement.

 The addition of two numbers represented in 2's complement is carried out by normal binary addition with carry discarded.

 The subtraction is carried out by taking 2's complement (B) of subtrahend and adding it to minuend (A).

 Overflow can be detected by inspecting last 2 carries out of addition by EX-OR them. If different then overflow is detected.

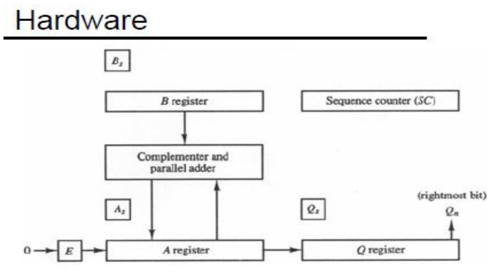
 For addition simply implement add then see overflow. For subtract add 2's complement of B to A and watch overflow since the A and –B could be of same sign.


Algorithm
Subtract
Add
Minuend in AC Subtrahend in BR
Augend in AC Addend in BR
$AC \leftarrow AC + \overline{BR} + 1$ $V \leftarrow$ overflow
$AC \leftarrow AC + BR$ $V \leftarrow$ overflow
END
END

Multiplication Algorithms
 The multiplication of two numbers in signed magnitude representation is carried out by successive shift and adds.

 Look at successive bits of multiplier (least significant bit first), if bit=1 multiplicand is copied else if bit=0, zero is copied down shifted one bit to left from previous copies. Finally all numbers are added and their sum forms the product.

Hardware Implementation



It is necessary to supply adder for summation two binary numbers and successively accumulate the partial products in a register.

 Instead of shifting multiplicand to the left the partial product is shifted to the right.

 When corresponding bit of multiplier is 0 no need to add zeros to partial product, just do nothing.

 Hardware for multiplication consist of complementer and parallel adder, Registers A and B with their sign bits As and Bs, multiplier Q register, and its sign in Qs. SC is set to number of bits in multiplier and when reaches zero it means partial addition has finished and product is ready in A.

 Multiplicand in A register while multiplier in Q register initially. The sum of A and B is the partial product in EA. A and Q registers are shifted together giving rise to new bit from Q in Qn bit which tells us if that bit is 0 or 1.
Algorithm
 Signs of Qs and Bs are compared then As and Qs are set as sign of the product. A and E bit are cleared with SC set to number of bits in Q

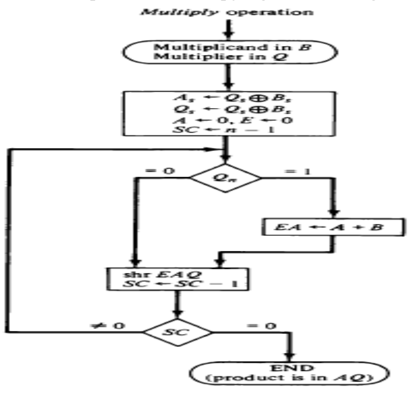
 Low order bit in Q is tested. If Qn=1 then B is added to A. if Qn=0 then nothing is done.

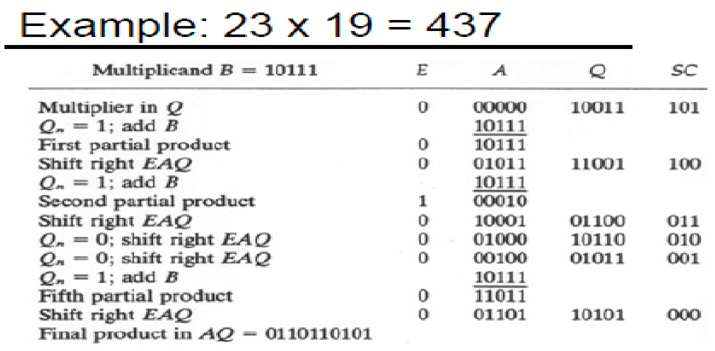
 Registers EAQ are shifted to the right.

 SC is decremented.

 Process stops when SC=0.

 Note that product is occupying A and Q registers and after each shift it replaces A and Q registers.





Example: 23 x 19 = 437

Multiplicand B = 10111	E	A	Q	SC
Multiplier in Q	0	00000	10011	101
$Q_n = 1$; add B		10111		
First partial product	0	10111		
Shift right EAQ	0	01011	11001	100
$Q_n = 1$; add B		10111		
Second partial product	1	00010		
Shift right EAQ	0	10001	01100	011
$Q_n = 0$; shift right EAQ	0	01000	10110	010
$Q_n = 0$; shift right EAQ	0	00100	01011	001
$Q_n = 1$; add B		10111		
Fifth partial product	0	11011		
Shift right EAQ	0	01101	10101	000
Final product in AQ = 0110110101				

Booth Multiplication Algorithm

 Gives procedure for multiplying binary integers in signed 2's complement representation.

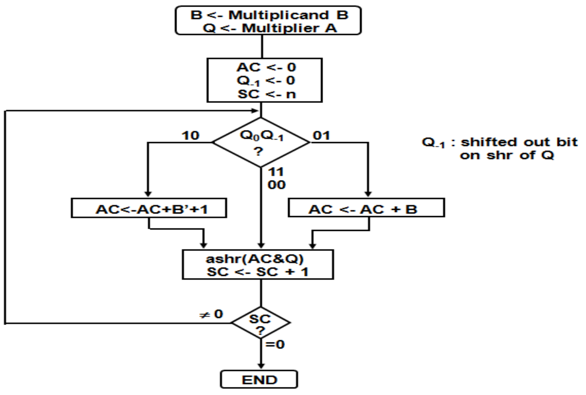
 It operates in fact that string of 0's in multiplier requires no addition but only shifting. And strings of 1's in multiplier needs addition by strings

 As with all multiplication schemes. Booth algorithm requires testing of multiplier bits and shifting of partial products. Before shifting, the multiplicand may be added to partial product, subtracted from partial product, or left unchanged.

o Multiplicand is subtracted from partial product when first least significant 1 of string of 1's in multiplier is encountered. (10)

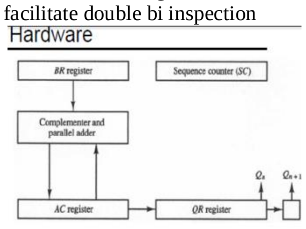
o Multiplicand is added to partial product when encountering first 0 and if there were a previous 1 in a string of 0's in multiplier. (01)

o The partial product does not change when multiplier bits are identical. (00 or 11)
The algorithm works for positive or negative multipliers in 2's complement representation.



The hardware implementation requires register configuration shown in next figure.

- We have BR register to hold multiplicand, QR holding multiplier, A register holding partial product with QR, and Qn+1 flip flop to facilitate double bi inspection



A numerical example of Booth algorithm is shown in next table for n=5. It shows steps of multiplying (-9) X (-13) = +117.

Q_0Q_{-1}	B = 10111 B'+1=01001	AC	Q	Q_{-1}	SC
	Initial	00000	10011	0	101
10	Subtract B	01001			
		01001			
	ashr	00100	11001	1	100
11	ashr	00010	01100	1	011
01	Add B	10111			
		11001			
	ashr	11100	10110	0	010
00	ashr	11110	01011	0	001
10	Subtract B	01001			
		00111			
	ashr	00011	10101	1	000

Array Multiplier

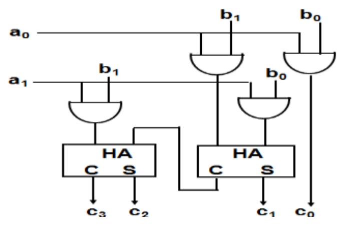
- The multiplication of 2 binary numbers can be done in one micro operation by means of combinational circuits that forms product bits all at once.

- This is a fast way to multiply numbers since it takes only the propagation delay.

- Next figure shows an example of multiplying 2 numbers each of 2 bits and a hardware implementation

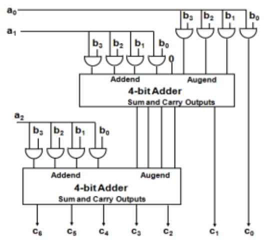
A = a_1a_0: Multiplier, B = b_1b_0: Multiplicand
C = B * A = $c_3c_2c_1c_0$

$$\begin{array}{cccc} & & b_1 & b_0 \\ & & a_1 & a_0 \\ \hline & & a_0b_1 & a_0b_0 \\ & a_1b_1 & a_1b_0 & \\ \hline c_3 & c_2 & c_1 & c_0 \end{array}$$



AND gates used to implement product of 2 bits.

- The first partial product formed by using 2 AND gates. The second partial product is formed by using another 2 AND gates but shifted one position to the left
- The 2 partial products are added using 2 Half Adder circuits.
- For j bits by k bits multiplications, we need j X k AND gates and (j-1) k-bit adder circuits to get (j + k) bits product.
- For 4 bits by 3 bits multiplication, next figure shows its implementation.
- Since k=4 and j=3 we have 12 AND gates and 2 4-bit Adders to produce a product of 7 bits



Memory Hierarchy

The memory unit that communicates with directly with CPU is called main memory.
o Not enough storage space.

o SRAM or DRAM

o Contains programs and data currently needed are stored here
 Devices that provide backup storage are called auxiliary memory.
o Most common are magnetic disks and tape drives.

o Used to store system programs, large data files, backup data

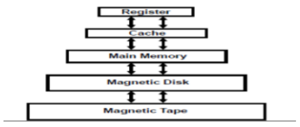
o Not urgently needed data are stored here.
 Total memory capacity of a computer can be visualized as a hierarchy of components.
o Ranges from 1- slow but high capacity to 2- relatively faster and less capacitive main memory to 3- smaller and fast cache memory.

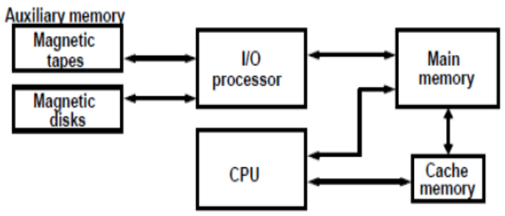
o At bottom of hierarchy you can find slow magnetic tapes(removable files)

o at middle you can find magnetic disks (backup storage)

o then main memory which can communicate with CPU and auxiliary memories.

o At top of pyramid resides the cache memory. Used to increase speed of processing. Compensate of speed difference between CPU and main memory. Usually in cache segments of programs and data frequently accessed are stored to benefit from high speed access by CPU not found in main memory





Main Memory
 Main memory is central storage unit in computers

 Fast and relatively large type of memory and used to store programs and data used in program execution

 RAM: integrated circuit chips (Static or Dynamic)
o SRAM consists of flip flops as storage media. Stored data remains valid as long as power is applied to the unit

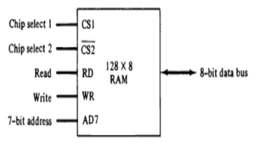
o DRAM stores data as form of electric charges in small capacitors. Capacitors are provided by CMOS transistors. Needs refreshing periodically as charges on small capacitor discharge soon (need electronic control unit for that).

o DRAM compared to SRAM offer reduced power consumption and larger capacity. But SRAM are faster.
 ROM is different type of main memories. Used to store programs and data that does not change at all (programs, tables, etc.)
o Used ROMs for storing bootstrap loaders
ROMs are used to startup any computer
ROM and RAM are available in different sizes. And usually we have to combine many chips to increase size.
 RAM chips consist of a number of address pins, bidirectional data pins, and some control pins.
o Bidirectional means data can be initiated from memory to the outside or from outside to memory chip. o Next figure shows 128 words of 8 bits per word RAM chip. This requires 7 address bits and 8 bidirectional data bits.

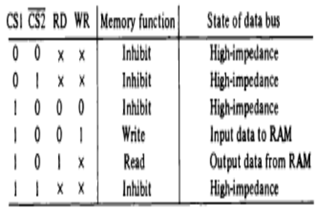
o Next figure shows a RAM chip that have 2 chip enables CS1=1 and CS2#=0. Those called chip select controls and they enable the chip to function if selected.

o Another 2 controls RD and WR. When R=1 then chip in a read status such that it provided data into data pins from location specified by address. If WR=1 then it is in write status and it accepts data from data bits into the chip's storage location specified by address

Figure 12-2 Typical RAM chip.



(a) Block diagram



CS1	$\overline{CS2}$	RD	WR	Memory function	State of data bus
0	0	x	x	Inhibit	High-impedance
0	1	x	x	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	x	Read	Output data from RAM
1	1	x	x	Inhibit	High-impedance

(b) Function table

ROM chips are constructed the same way. It has address bits, one direction data bits, CS control pin and/or RD control pin.

o Next figure shows 512 words ROM chip each with 8 data bits. See how address pins and storage locations are related.

Internal binary cells of ROM occupy much less space than RAM.

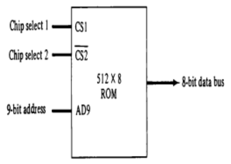


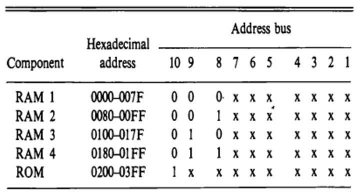
Figure 12-3 Typical ROM chip.

Memory Address Map and Connection to CPU

 Memory Address Map is “address space assignment to each memory chip”.

 Assume computer system with 512 bytes of RAM and 512 bytes of ROM as shown in next figure. The memory address map is shown in next table.

TABLE 12-1 Memory Address Map for Microprocomputer



Component	Hexadecimal address	Address bus 10	9	8	7	6	5	4	3	2	1
RAM 1	0000–007F	0	0	0	x	x	x	x	x	x	x
RAM 2	0080–00FF	0	0	1	x	x	x	x	x	x	x
RAM 3	0100–017F	0	1	0	x	x	x	x	x	x	x
RAM 4	0180–01FF	0	1	1	x	x	x	x	x	x	x
ROM	0200–03FF	1	x	x	x	x	x	x	x	x	x

RAM chips are 128 words each so they need 7 address lines.

o ROM chips are 512 words each so they need 9 address lines.

o For RAM selection A10 is always 0 whereas for ROM selection A10 is always 1.

o Memory chips are connected to CPU through address and data busses. RAM chips stores 128 by 4 = 512 bytes. ROM chip stores 512 bytes alone.

o A10 selects ROM chip if it is 1. And it selects all RAM chips if it is 0.

o RD and WR control pins are connected to RD and WR pins in corresponding ROM and RAM chips to initiate read or write operation from CPU.

o Address range from 0 to 511 is assigned for RAM chips while address range from 512 to 1023 is assigned to ROM chip

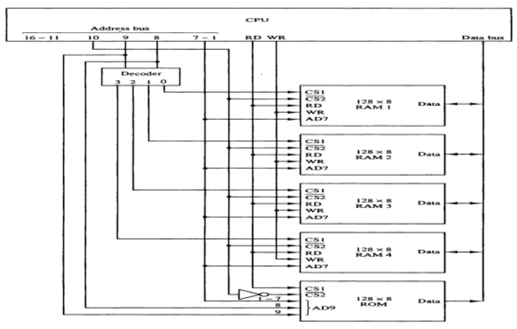


Figure 12-4 Memory connection to the CPU.

Auxiliary Memory

 Common used secondary memory is magnetic disks and magnetic tapes.

 Other devices (magnetic drums, bubble memory, CD, DVD, flash disks, etc.)

 The important characteristics of those storage devices are

o Access mode

o Access time

o Transfer rate

o Capacity

o Cost.

 Access time

o Average time needed to reach storage location and obtain contents.

 Access time = seek time + transfer time

 Seek time:

o Transfer time: time required to transfer data to-from device.

 Secondary storage devices are organized into records (blocks). Reading or writing is always done with entire record.

 Transfer rate

o The number of characters or words the device can transfer in one second.

MAGNETIC DISKS

 Circular plate constructed of metal or plastic coated with magnetized material.

 Both sides of disks are used and severatemsl sysl disks may be stacked with read-write heads available for each surface.

 All disks rotate together at high speed

 Bits are stored in tracks which are concentric circles

 Tracks are divided into sections called sectors

 Some disk systems use single read-write head moveable to different tracks using mechanical assembly

 Others use multiple read-write heads positioned on each track (faster , more expensive).

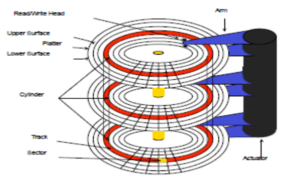
 Addressing used to specify disk number, surface, track number, and sector within track.

 After head positioned at track, must wait to synchronize with sector

 Then reading data will start as same speed of rotation

 Hard disks are permanently attached to unit and cannot move. Floppy disks are removable ones. With 2 sizes 5.25 and 3.5

Disk Structure



Magnetic tapes

Strip of plastic coated with magnetic recording material.

Bits are recorded as magnetic spots along several parallel tracks(7 to 9 tracks to form character with parity).

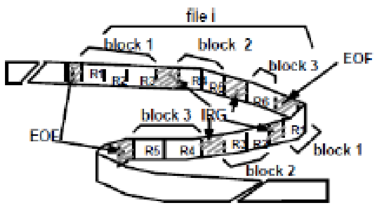
Read-write heads are positioned on each track.

Magnetic tape units can be started stopped, forward moved, or reverse moved or rewound.

Data are recorded in records (number of characters) followed by gaps between record for synchronization.

At start and end of each record there is ID bit patterns.

Records are identified by reading ID bit patterns.



Associative Memory

 Accessed by the content of the data rather than by an address. Also called Content Addressable Memory (CAM).

 When word is written to CAM, no address is needed; next available unused storage location is located. When word is read from CAM, the content of word or part of it is specified, the memory locates all words which give match and marks them for reading.

 Associative memories are expensive and used for application where time search is critical.
HARDWARE ORANIZATION

- Consists of memory array of m words each of n bits, argument register A and key register K each of n bits.
- Match register M has m bits, one for each memory word
- Each word of memory is compared in parallel with content of argument register and set corresponding bit in match register.
- Those bits set in match register indicate their words has match.
- Key register provides mask to select particular bits in argument word to be included in match or not.
- 1 means corresponding bit in argument register is in match and 0 means not.

Figure 12-6 Block diagram of associative memory.

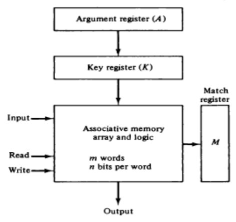
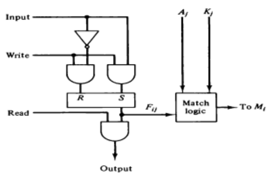
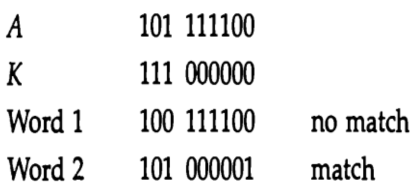


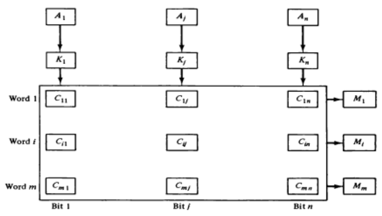
Figure 12-8 One cell of associative memory.





A	101 111100	
K	111 000000	
Word 1	100 111100	no match
Word 2	101 000001	match

Figure 12-7 Associative memory of m word, n cells per word.



MATCHING LOGIC

 The match logic for each word can be derived from comparison algorithm of two binary numbers.

1. K is neglected

Word i is equal to argument A if Aj = Fij for j=1 ,2, 3, …,n
xj = Aj Fij + A’j F’ij
for word i to be equal to argument A we must have all xi variables equal 1. The Boolean function for that will be
Mi = x1 x2 x3 x4 ……. xn
2. K is included
Requirement will be
xi + K’j = xi if Kj=1
= 1 if Kj=0
Then match logic will be
Mi = (x1 + K’1) (x2 + K’2) (x3 + K’3) ….. (xn + K’n)

The function now can be expressed in detail as
Mi = Π(Aj Fij + A’jF’ij + K’j) for j=1 to n

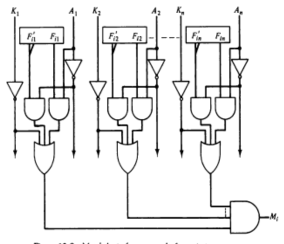


Figure 12-9 Match logic for one word of associative memory.

Cache Memory

 Analysis has shown that references to memory at given interval of time is confined within few localized areas in memory. (Locality of Reference)

 Programs has loops and repeated subroutine calls encountered frequently. So computer repeatedly refers to set of instructions in memory of the loop

 Same applies for subroutine; every time a subroutine is called the instructions of teh subroutine will be executed.

 Memory reference of data also tend to be localized . lookup tables data repeatedly refer to same area in memory.

"Temporal Locality"
 The information which will be used in near future is likely to be in use already(e.g. Reuse of information in loops)
"Spatial Locality"
 If a word is accessed, adjacent(near) words are likely accessed soon
o (e.g. Related data items (arrays) are usually stored together
instructions are executed sequentially

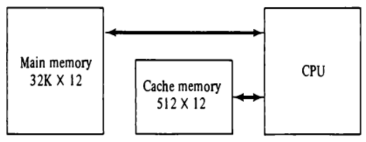


Figure 12-10 Example of cache memory.

"Cache Memory"
 The property of Locality of Reference makes the Cache memory systems work

 Cache is a fast small capacity memory that should hold those information which are most likely to be accessed.

 Then the average memory access time can be reduced dramatically

 Placed between processor and main memory

 Have access time less than main memory access time by a factor of (5 – 10)

"Cach and memory Access"
 All the memory accesses are directed first to Cache. If the word is in Cache; Access cache to provide it to CPU.

 If the word is not in Cache; bring a block (or a line) including that word to replace a block now in Cache. Block varies between (1 – 16 words)

"Cache Memory system Performance"
Hit Ratio : "% of memory accesses satisfied by Cache memory system"
If processor searches for a word in cache and finds it then we call this state as "hit"; in the same way, if the word is not found in cache then must it be pulled from main memory and placed in cache, then we call this state as "miss" ,Te = Tc + (1 -h) Tm

Where: Te: Effective memory access time in Cache memory system ,Tc: Cache access time Tm: Main memory access time

Example: Assume a cached processor system with cache access time of Tc = 0.4 us, and a memory access time of Tm = 1.2us, and cache hit ratio of h = 0.85%. Then get the effective access time? Te = 0.4 + (1 -0.85) * 1.2 = 0.58 us
<u>"Mapping Function: Associative Mapping"</u>
 Any block location in Cache can store any block in memory
o Most flexible

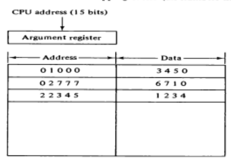
 Mapping Table is implemented in an associative memory
o Fast, very Expensive

o Stores both address and the content of the memory word.

- Cache controller search for the existence of the address part. If found then content is accessed; otherwise the main memory is accessed and the address-data pair is placed in next available place in cache.

If cache is full then a decision must be taken to which line in cache to be replaced by the new content (need a replacement algorithm).

Figure 12-11 Associative mapping cache (all numbers in octal).



"**Mapping Function: Direct Mapping"**

- Associative memories are expensive compared to RAMs

- Each memory block has only one place to load in Cache

- Mapping Table is made of RAM instead of CAM

- n-bit memory address consists of 2 parts; (k) bits of Index field and (n-k) bits of Tag field

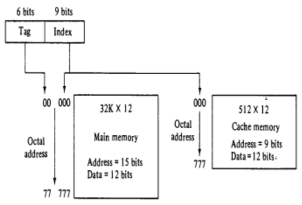
- n-bit addresses are used to access main memory and k-bit Index is used to access the Cache. Tag part of the address is stored with the data in cache line that it represent.

location 00000 will be placed in cache in location 000. And its content 1220 will be placed in cache at that address

content of 02777 which is 6710 will be placed in address 777 in cache.

the next main memory addresses 00000, 01000, and 02000 when referenced will occupy the same cache line address of 000. But they have different TAG parts.

Figure 12-12 Addressing relationships between main and cache memories.



Direct mapping cache organisation

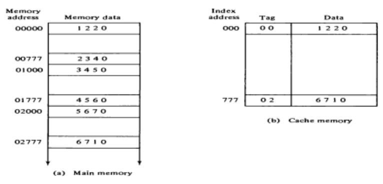


Figure 12-13 Direct mapping cache organization.

Operation of Direct mapped caches

- address is generated from CPU with 2 parts INDEX and TAG
- cache is accessed using INDEX; the required word is accessed and TAG if compared with cache TAG
- IF matches then it is a "Hit"

o Data is pulled from this line to processor

- If not a match then it is a "Miss"

o The required address content is read into cache. ?It may replace an old content with the same INDEX but not the TAG

o A copy is presented to the processor

Example: block size = 8 bytes

- Each line in cache stores TAG and DATA
- Data will be content of 8 bytes addressed sequentially in main memory
- We have 64 cache lines
- TAG stored with data ion cache line is the remaining address of memory location after the 3 bits word and 6 bits block

Disadvantage

Hit ratio may drop if 2 or more address words whose address has the same INDEX but different TAG are accessed repeatedly.

- 2 main memory address can have the same index part but different tag part will occupy the same cache line

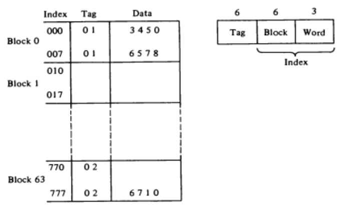


Figure 12-14 Direct mapping cache with block size of 8 words.

Cache Write policies
Write Through

- Memory is always updated
- When writing into memory
  - o If Hit, then both cache and memory is written in parallel
  - o If Miss, then memory is written
  - o For a read miss, missing block may be overloaded onto a cache block
- Slow as memory is always accessed

Write-Back (Copy-Back)

- When writing into memory
  - o If Hit, only Cache is written
  - o If Miss, missing block is brought to cache and write into Cache
- For a read miss, replaced block must be written back to the memory. And this is the only time the block is updated.
- Memory is not up-to-date, i.e., the same item in cache and memory may have different value

<u>Reference</u>

1.‘Computer System Architecture’, Morris M. Mano, 3rd edition,Prentice Hall India.

2. Computer Organization and Achitecture, William Stallings ,8th edition,PHI

3. Computer Organization, Carl Hamachar, Vranesic, McGraw Hill.