MEMORY HIERARCHY

Memory Hierarchy

- Registers
  - In CPU
- Primary Memory
  - May include one or more levels of cache
  - “RAM”
- Secondary memory
  - Magnetic/Optical

CACHE:

-High Speed Memory (SRAMs)

-Small in Size

-High cost

MAIN MEMORY

-High density (DRAMs)

-Low cost

-Slower than Cache.

RAM

Static RAM

- Bits stored as on/off switches
- No charges to leak
- No refreshing needed when powered
- More complex construction
- Larger per bit
- More expensive
- Does not need refresh circuits
- Faster
- Cache

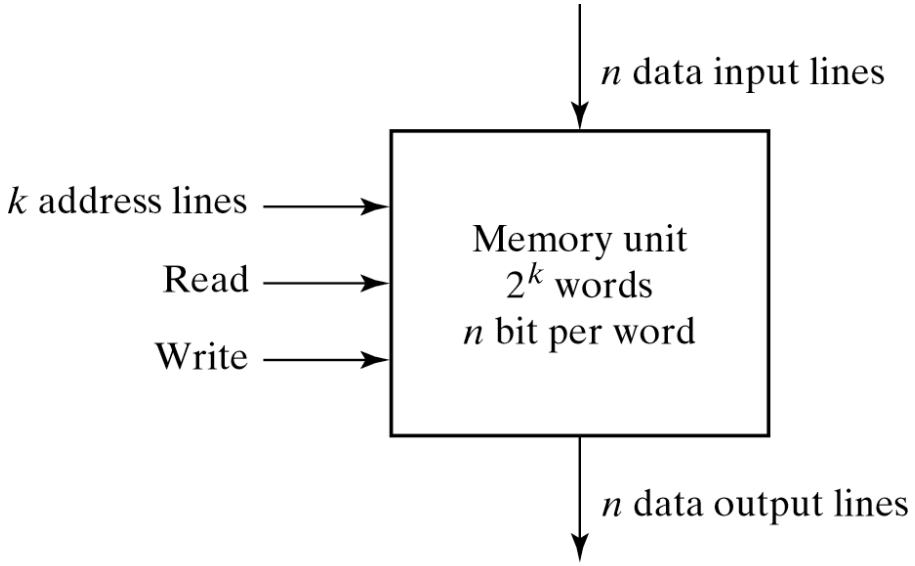
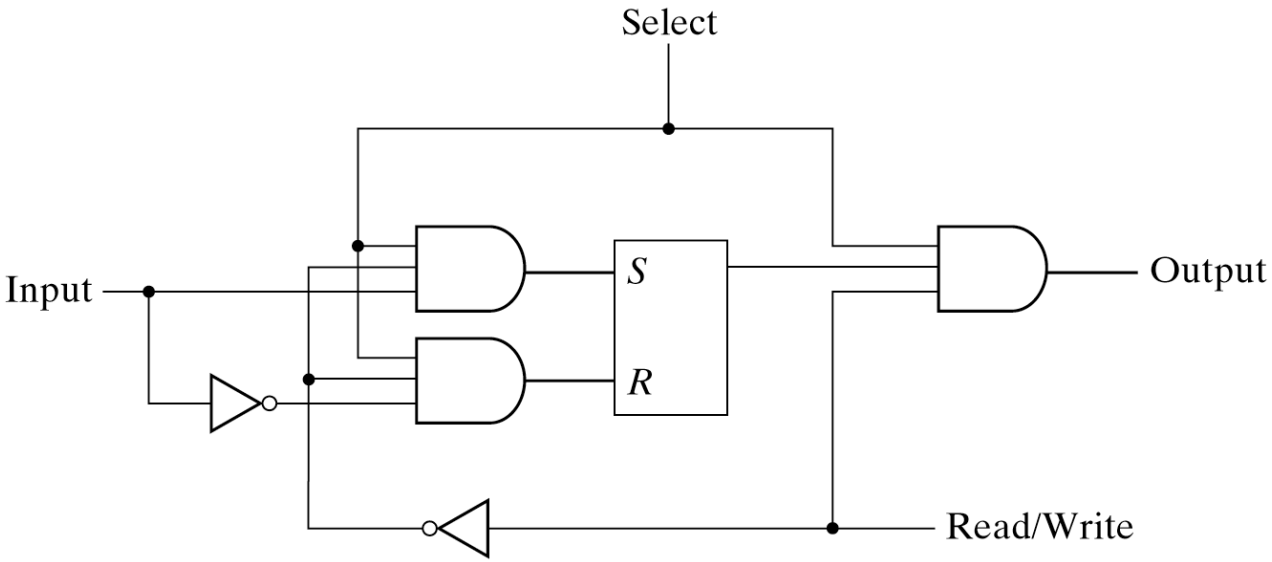


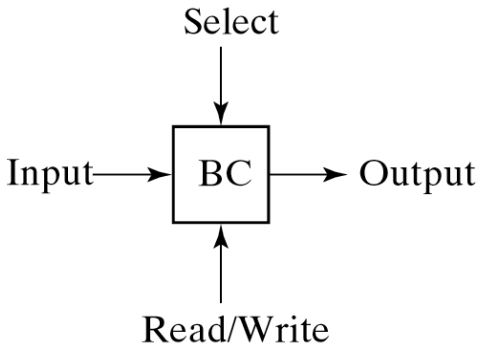
Fig. 7-2 Block Diagram of a Memory Unit

Memory address		
Binary	decimal	Memory contest
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
⋮	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

Fig. 7-3 Content of a 1024 × 16 Memory



(a) Logic diagram



(b) Block diagram

Fig. 7-5 Memory Cell

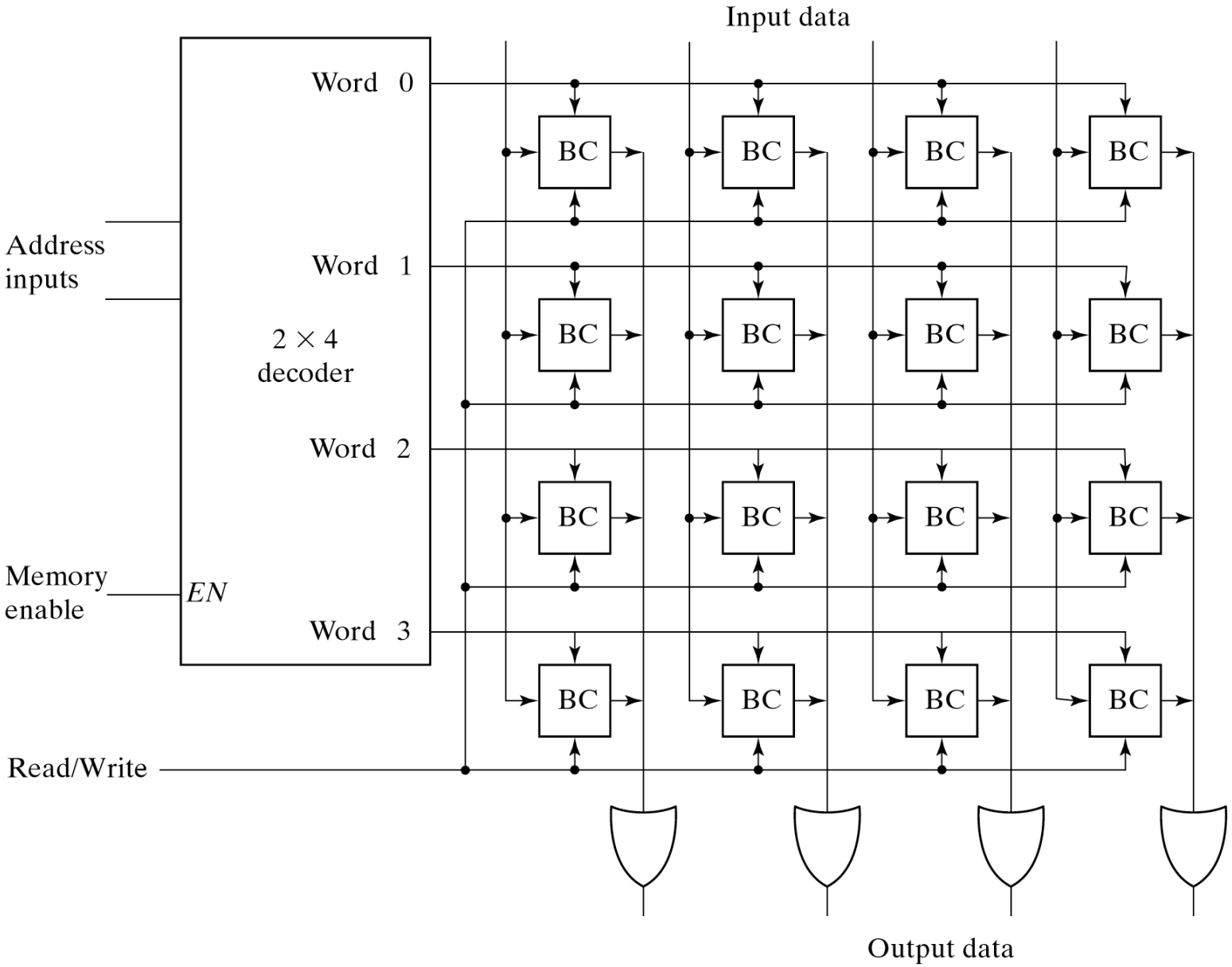
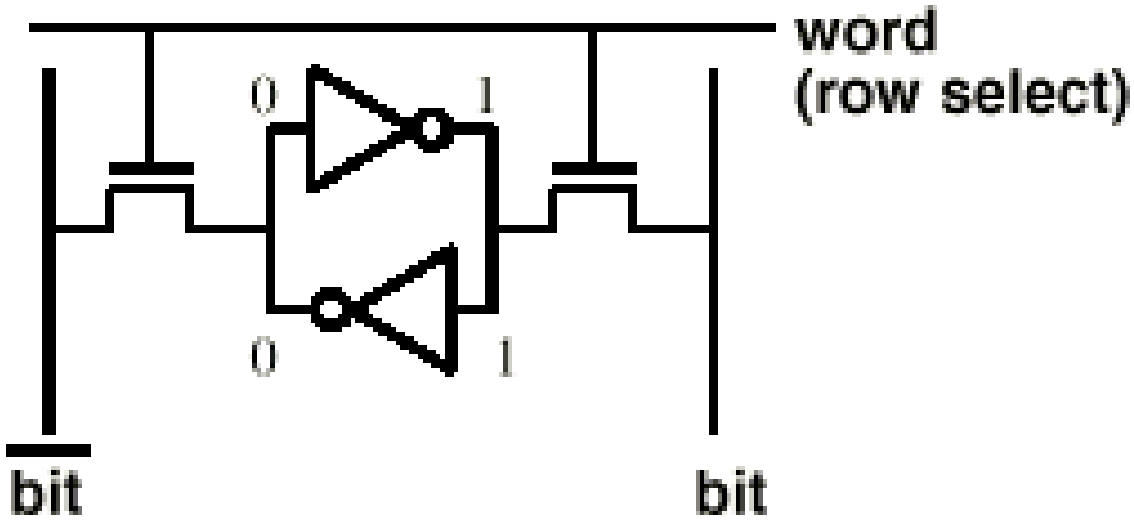
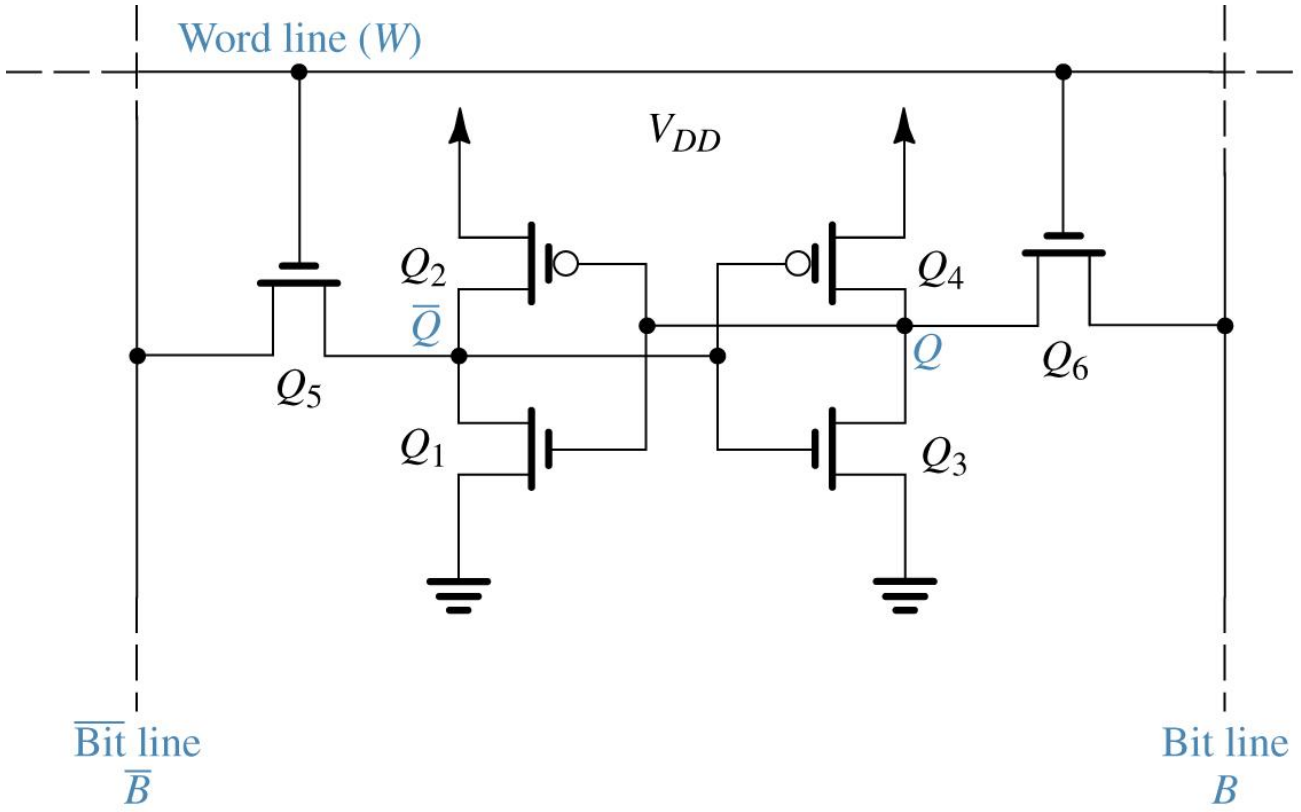


Fig. 7-6 Diagram of a 4 × 4 RAM

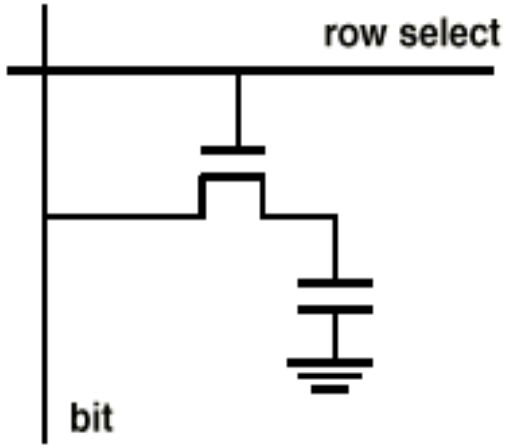
Static RAM Cell

6-Transistor SRAM Cell

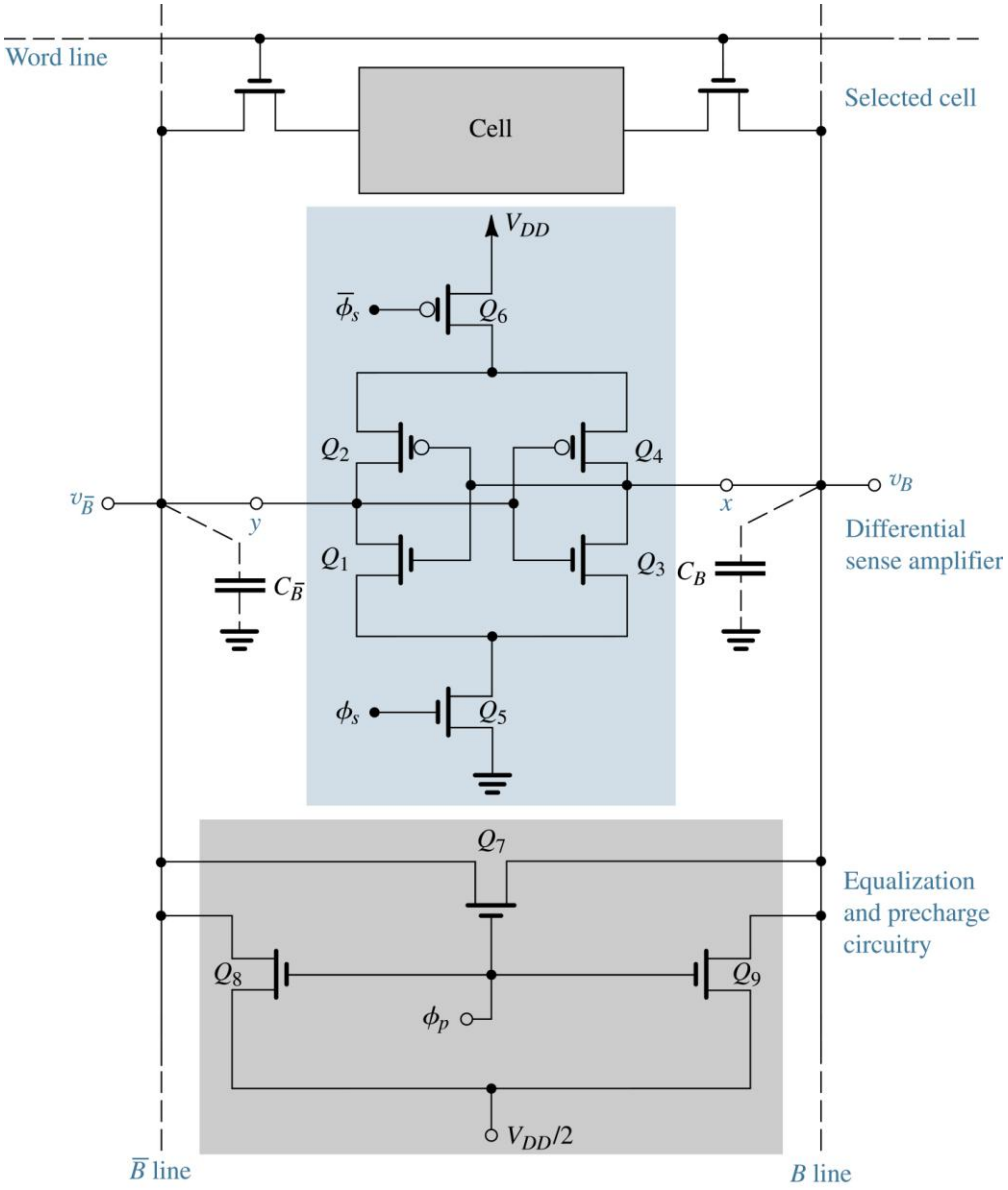


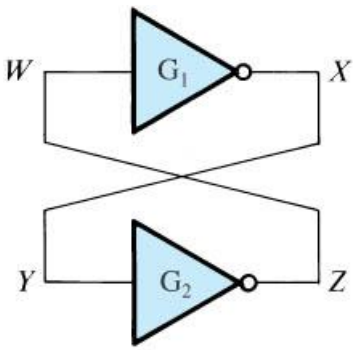

Word line (W)
V_{DD}
Q_2
$\overline{Q}$
Q_4
Q
Q_5
Q_6
Q_1
Q_3
$\overline{\text{Bit line}}$
$\overline{B}$
Bit line
B

1-Transistor Memory Cell (DRAM)

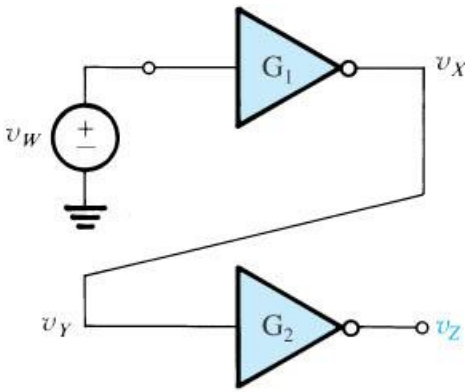


- Write:
  1. Drive bit line
  2. Select row
- Read:
  1. Precharge bit line to Vdd
  2. Select row
  3. Cell and bit line share charges
     Very small voltage changes on the bit line
  4. Sense (fancy sense amp)
  5. Write: restore the value
- Refresh
  1. Just do a dummy read to every cell.

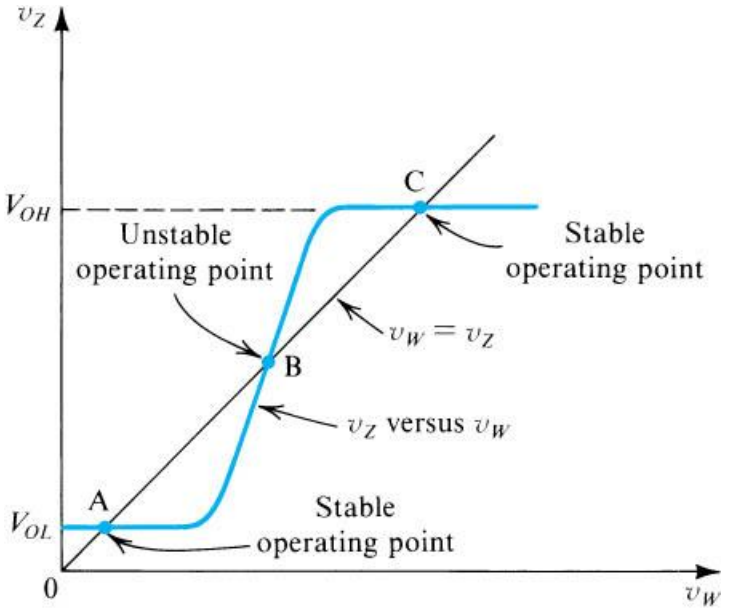

Word line
Cell
Selected cell
V_{DD}
$\overline{\phi}_s$
Q_6
Q_2
Q_4
$v_{\overline{B}}$
y
x
v_B
Differential sense amplifier
Q_1
Q_3
$C_{\overline{B}}$
C_B
ϕ_s
Q_5
Q_7
Equalization and precharge circuitry
Q_8
Q_9
ϕ_p
$V_{DD}/2$
$\overline{B}$ line
B line


W
G1
X
Y
G2
Z

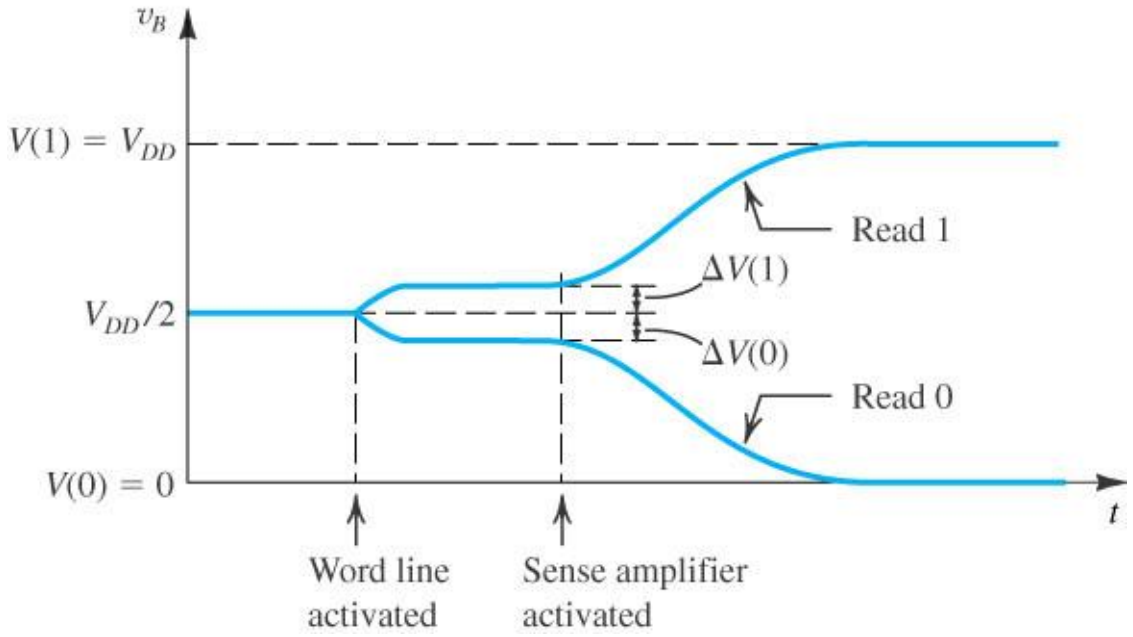
(a)

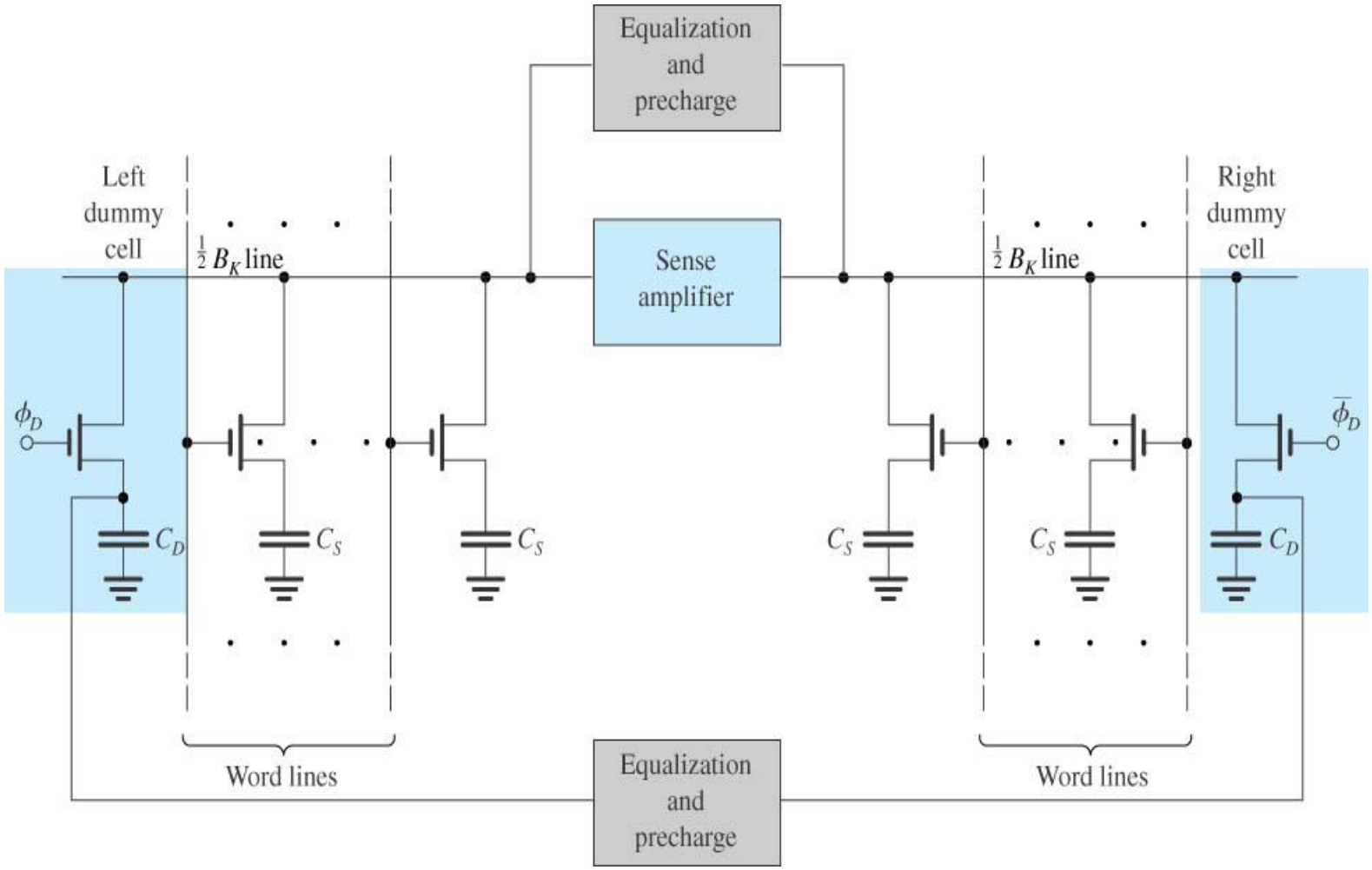

G1
vX
vW
+
−
vY
G2
vZ

(b)


vZ
VOH
C
Unstable
operating point
Stable
operating point
vW = vZ
B
vZ versus vW
A
VOL
Stable
operating point
0
vW

(c)


v_B
$V(1) = V_{DD}$
$V_{DD}/2$
$V(0) = 0$
Read 1
$\Delta V(1)$
$\Delta V(0)$
Read 0
t
Word line activated
Sense amplifier activated


Equalization and precharge
Left dummy cell
$\frac{1}{2}B_K$ line
Sense amplifier
$\frac{1}{2}B_K$ line
Right dummy cell
ϕ_D
$\overline{\phi}_D$
C_D
C_S
C_S
C_S
C_S
C_D
Word lines
Equalization and precharge
Word lines

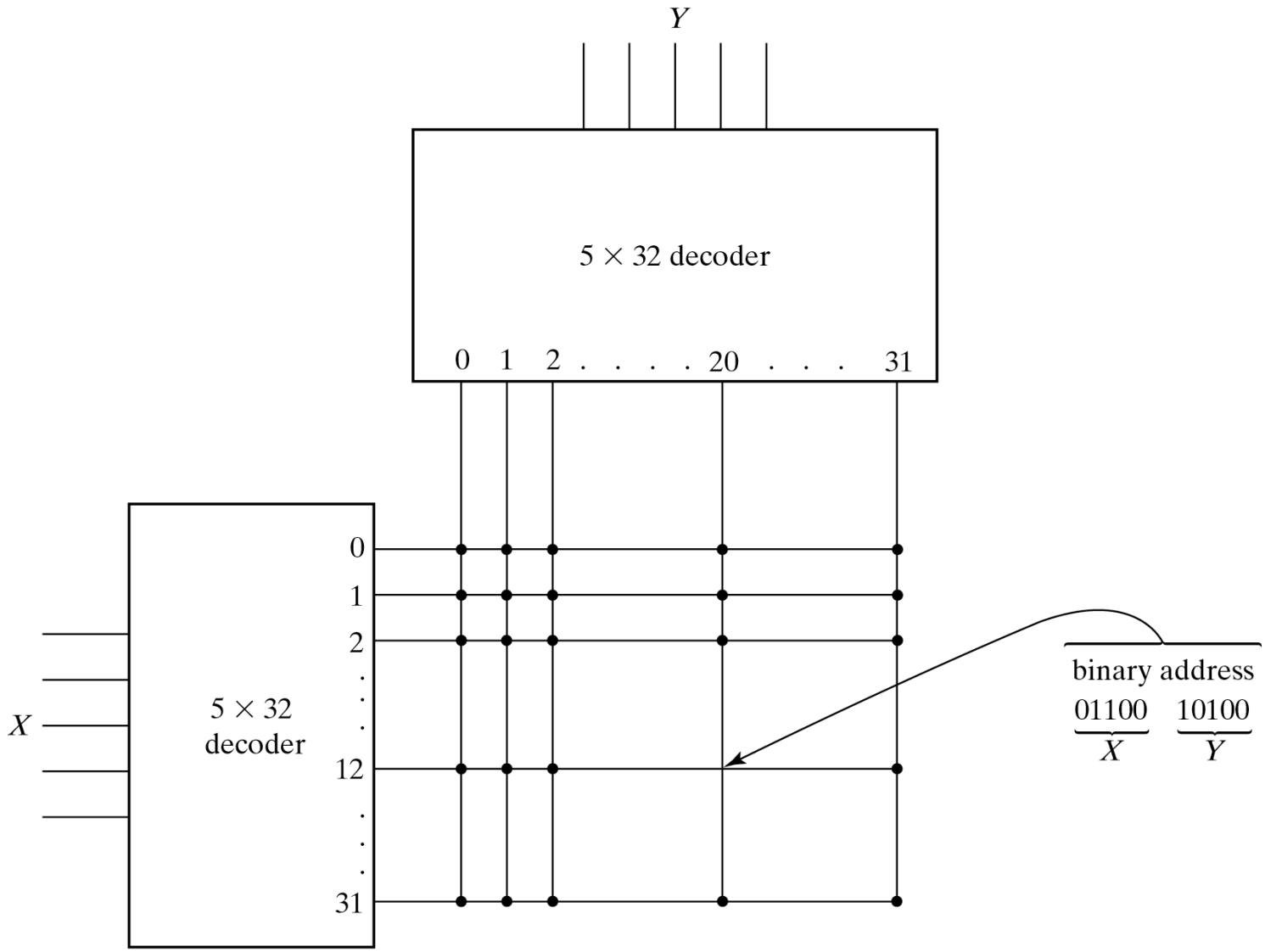


Fig. 7-7 Two-Dimensional Decoding Structure for a 1K-Word Memory

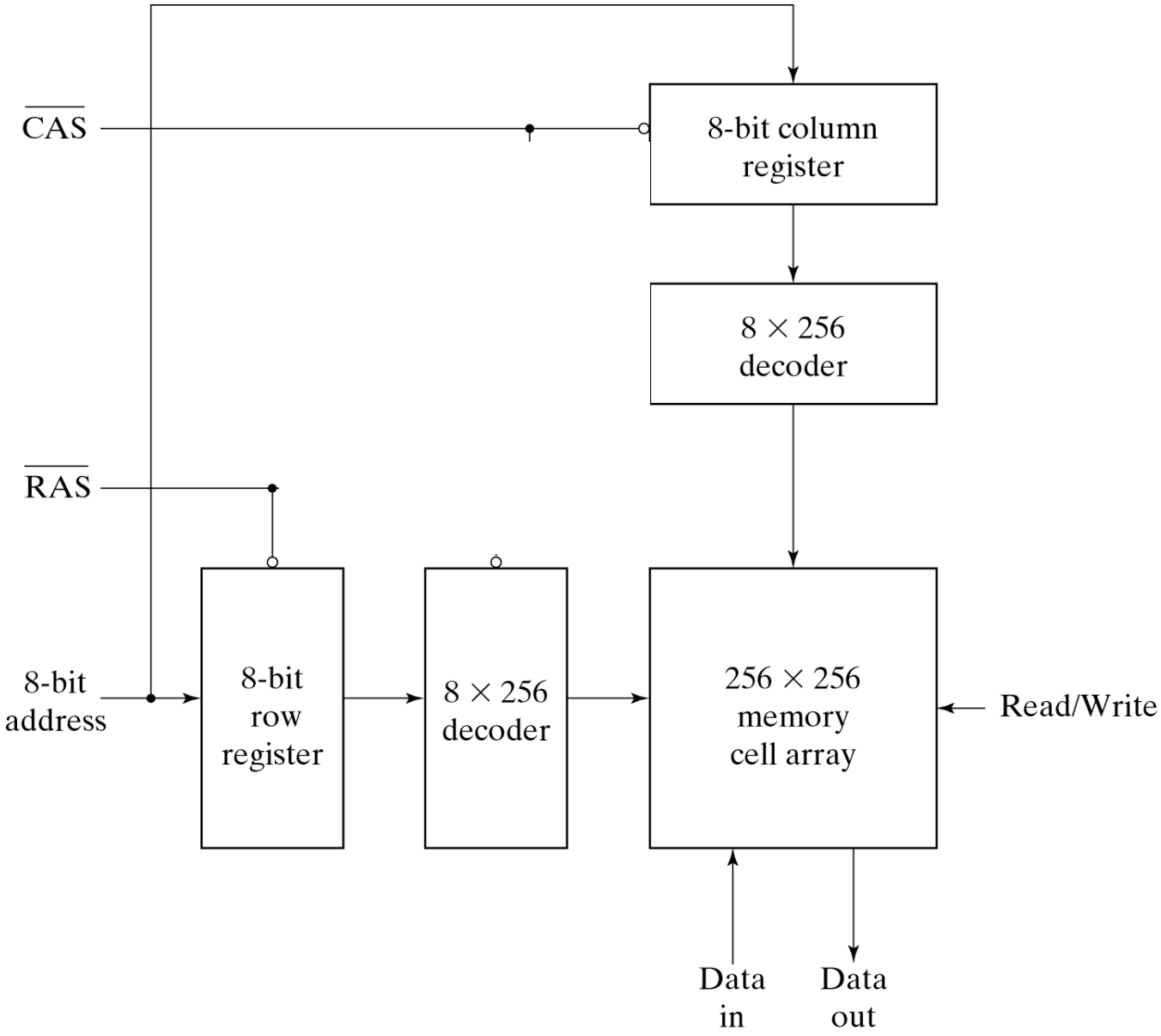
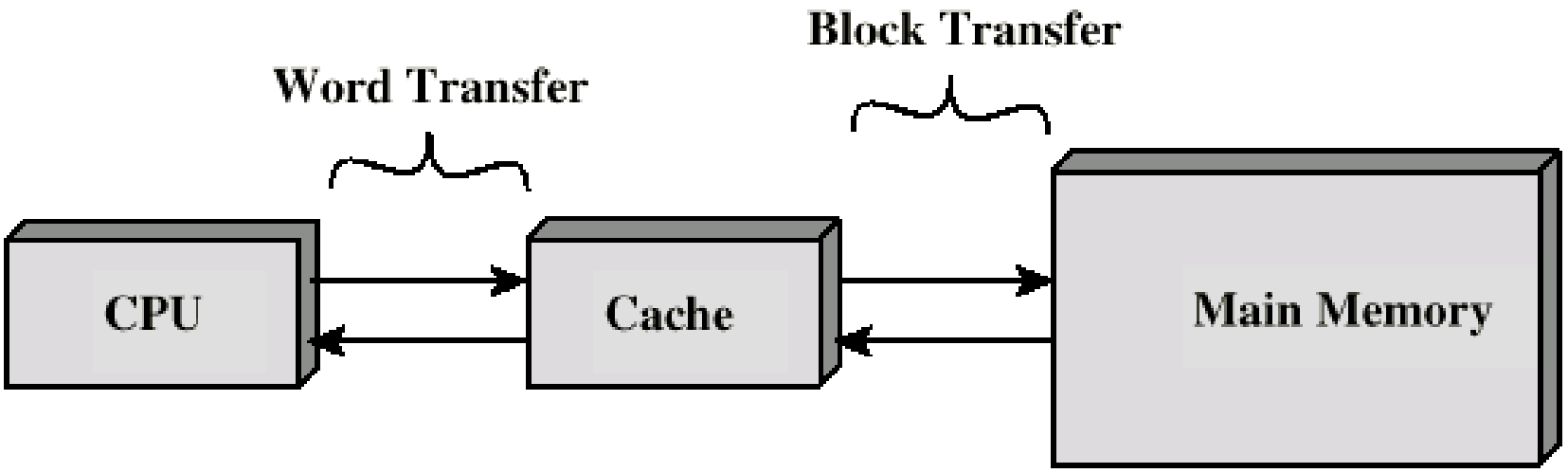


Fig. 7-8 Address Multiplexing for a 64K DRAM

Cache

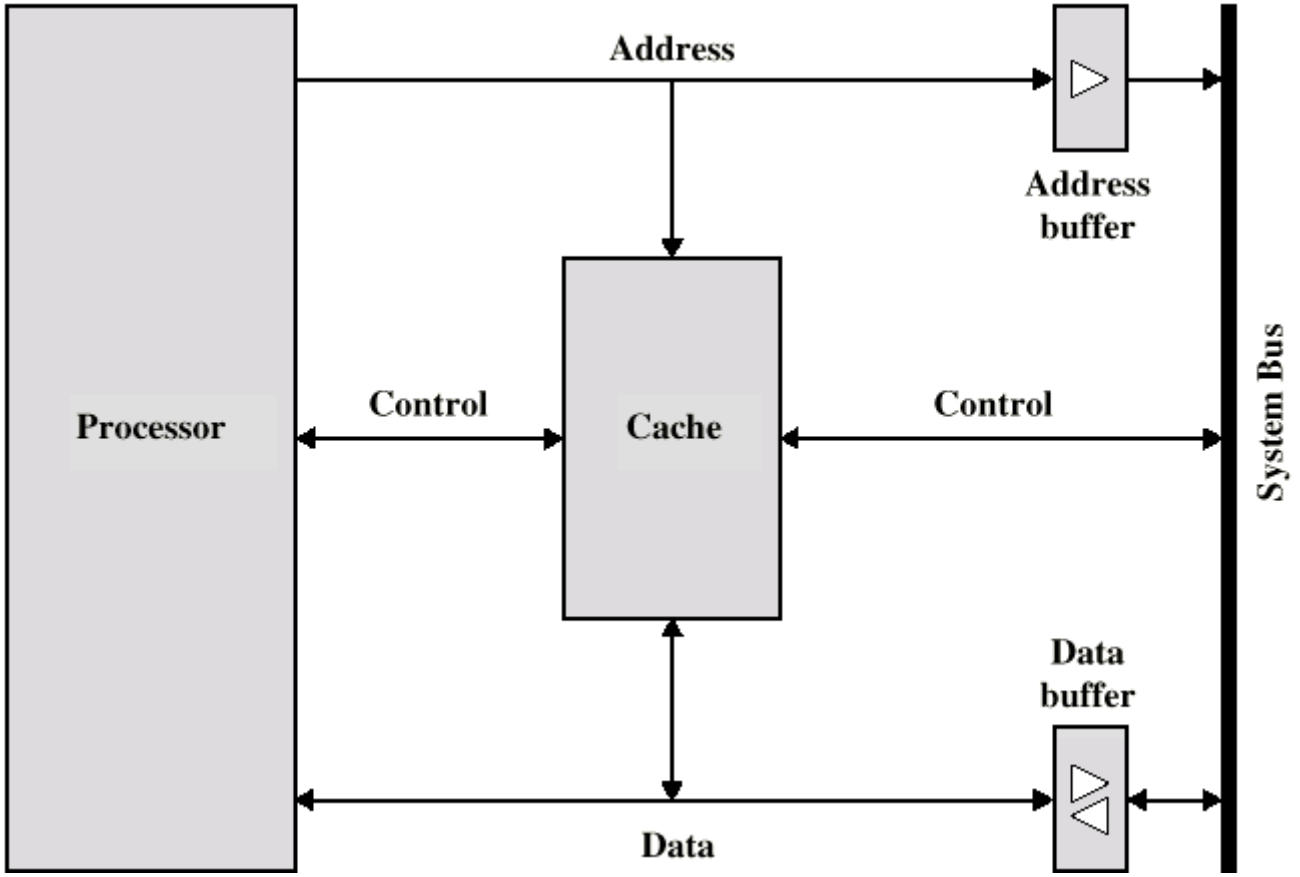
- Small amount of fast memory
- Sits between normal main memory and CPU



Cache operation - overview

- CPU requests contents of memory location
- Check cache for this data
- If present, get from cache (fast)
- If not present, read required block from main memory to cache
- Then deliver from cache to CPU
- Cache includes tags to identify which block of main memory is in each cache slot

Typical Cache Organization



Mapping Function

- Cache of 64kByte
- Cache block of 4 bytes
  - i.e. cache is 16k (2^{14}) lines of 4 bytes
- 16MBytes main memory
- 24 bit address
  - (2^{24}=16M)

Direct Mapping

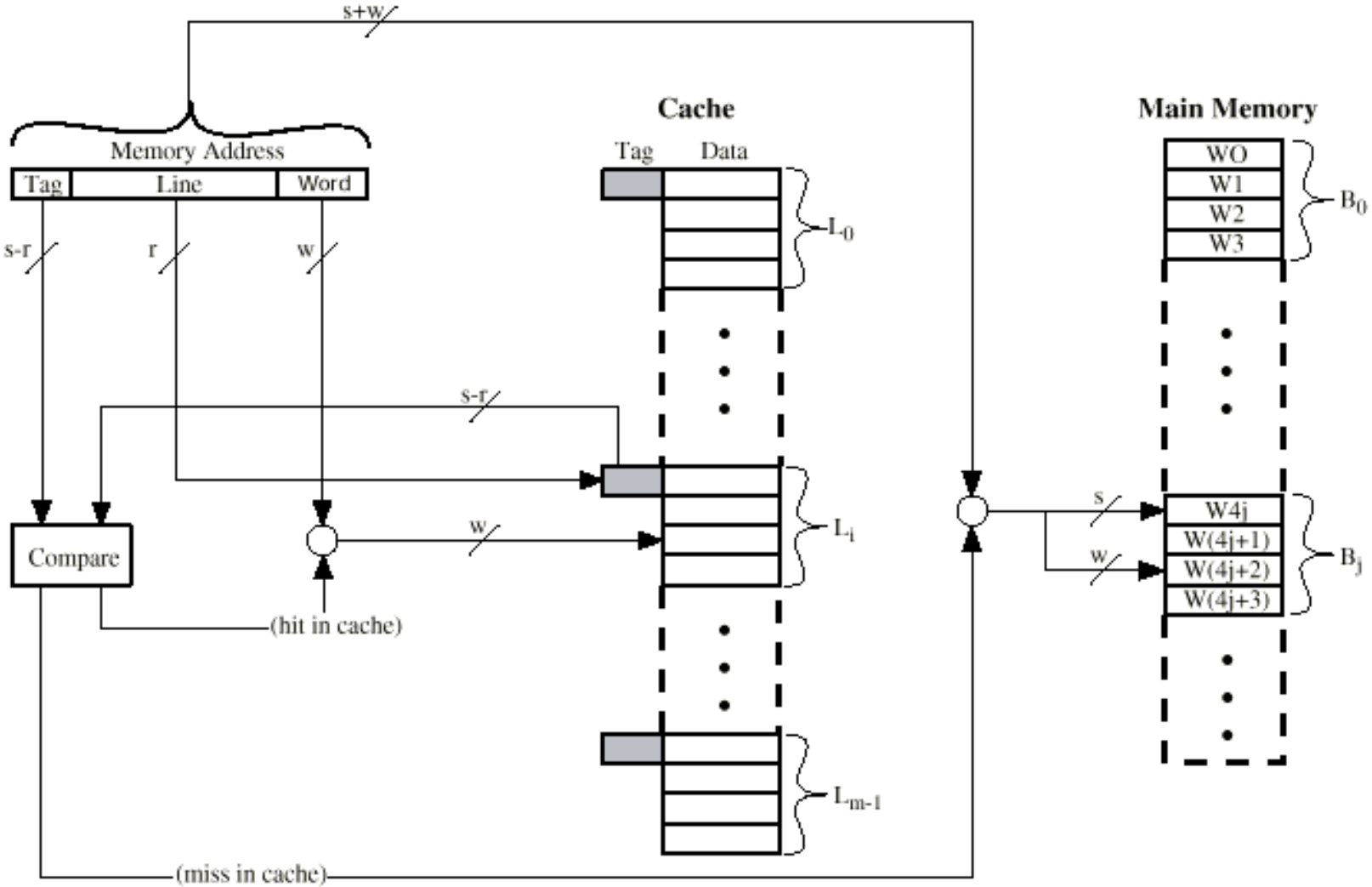
- Each block of main memory maps to only one cache line
  - i.e. if a block is in cache, it must be in one specific place
- Address is in two parts
- Least Significant w bits identify unique word
- Most Significant s bits specify one memory block
- The MSBs are split into a cache line field r and a tag of s-r (most significant)

Direct Mapping Address Structure

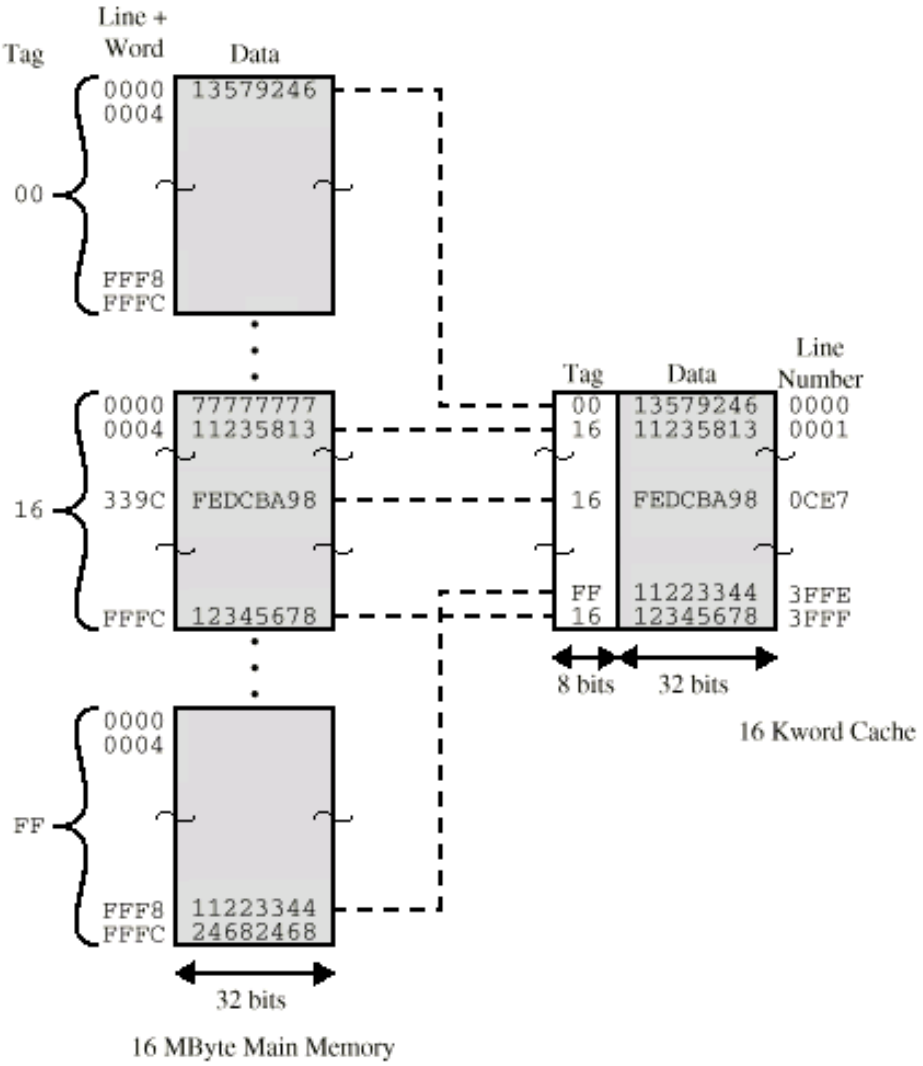
Tag s-r	Line or Slot r	Word w
8	14	2

- 24 bit address
- 2 bit word identifier (4 byte block)
- 22 bit block identifier
  - 8 bit tag (=22-14)
  - 14 bit slot or line
- No two blocks in the same line have the same Tag field
- Check contents of cache by finding line and checking Tag

Direct Mapping Cache Organization



Direct Mapping Example



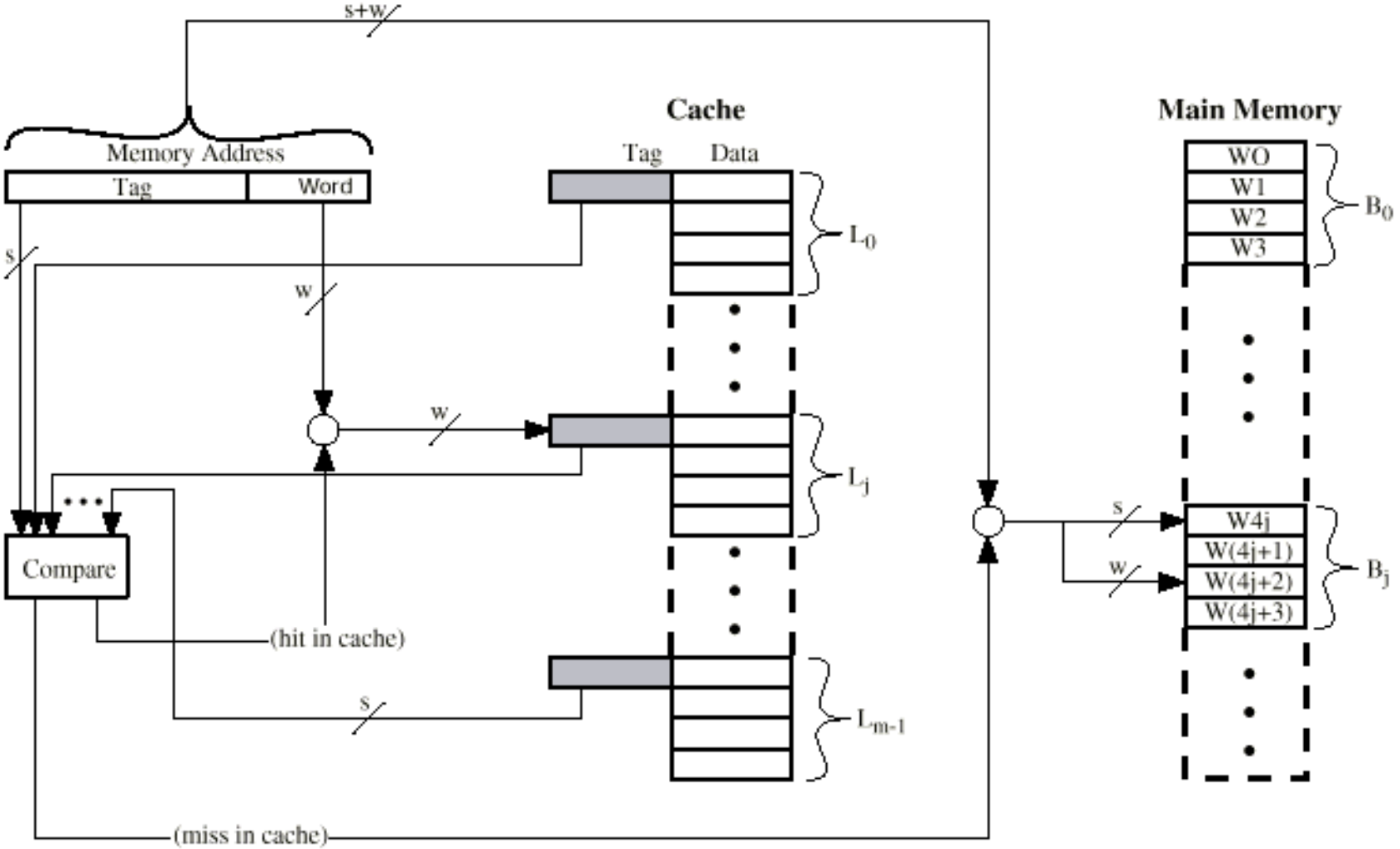
Direct Mapping pros & cons

- Simple
- Inexpensive
- Fixed location for given block
  - If a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high

Associative Mapping

- A main memory block can load into any line of cache
- Memory address is interpreted as tag and word
- Tag uniquely identifies block of memory
- Every line’s tag is examined for a match
- Cache searching gets expensive

Fully Associative Cache Organization

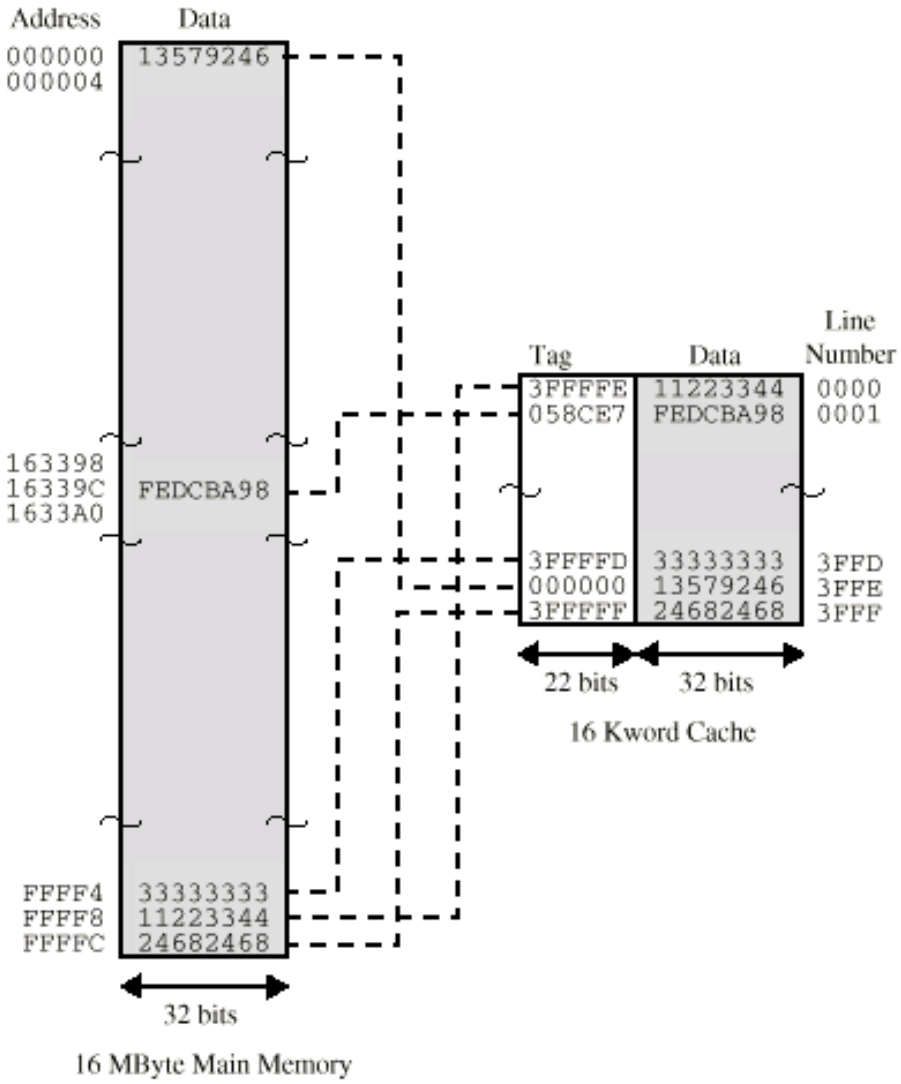


Associative Mapping Address Structure

Tag 22 bit	Word 2 bit

- 22 bit tag stored with each 32 bit block of data
- Compare tag field with tag entry in cache to check for hit

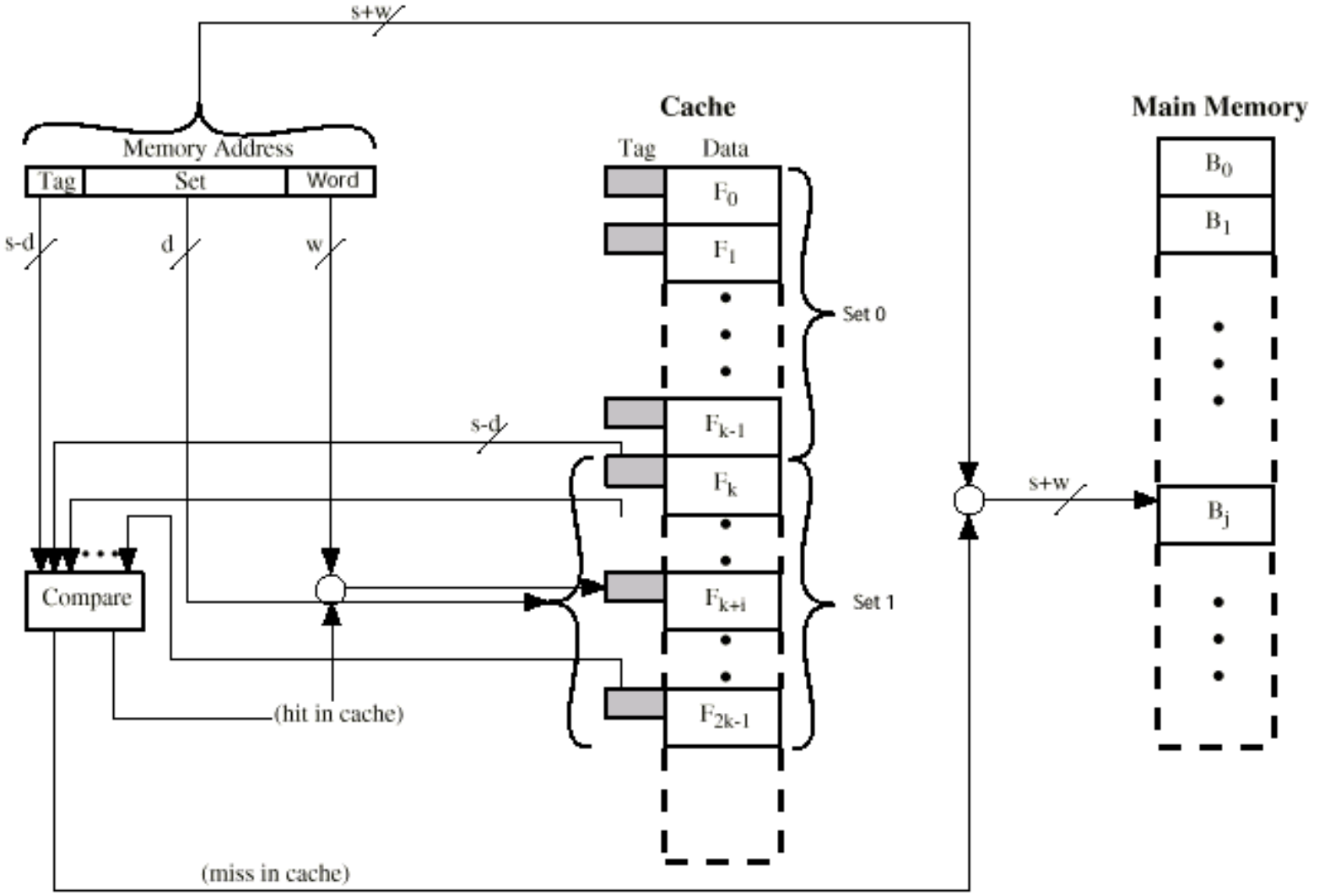
Associative Mapping Example



Set Associative Mapping

- Cache is divided into a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
  - e.g. Block B can be in any line of set i
- e.g. 2 lines per set
  - 2 way associative mapping
  - A given block can be in one of 2 lines in only one set

K- Way Set Associative Cache Organization

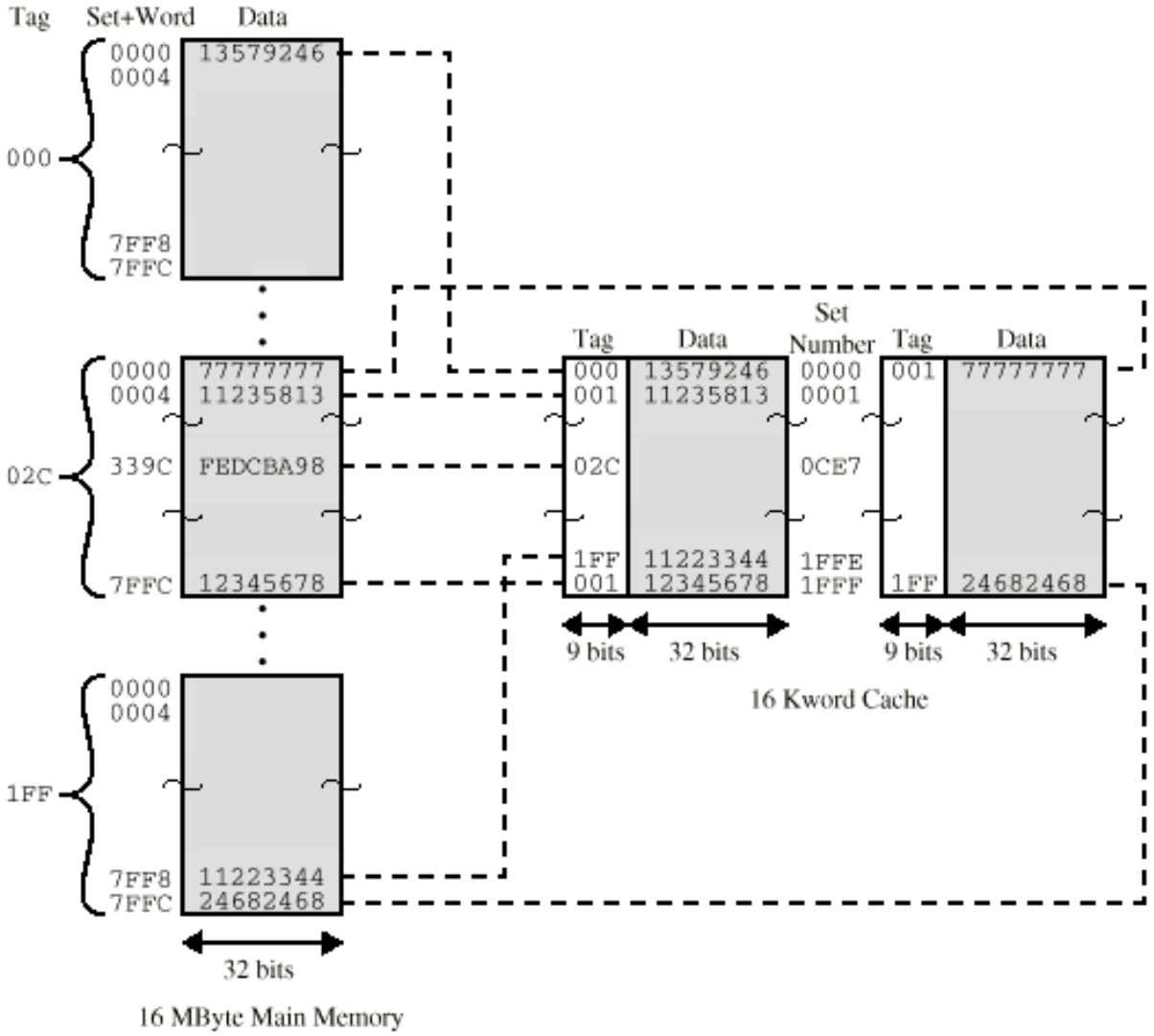


Set Associative Mapping Address Structure

Tag 9 bit	Set 13 bit	Word 2 bit

- Use set field to determine cache set to look in
- Compare tag field to see if we have a hit

Two Way Set Associative Mapping Example



Main Memory 64kB

Block size 8 bytes

Direct Mapped Cache – 32 lines

How is the 16 bit mem add divided

Into what line a byte with add below be stored

0001 0001 0001 1011

1100 0011 0011 0100