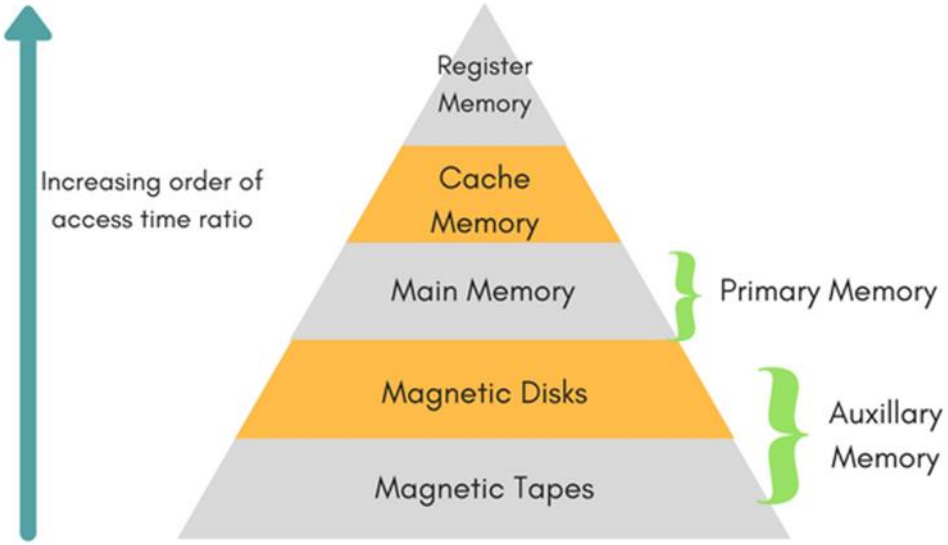
Chapter Two

Main Memory & Addressing Modes

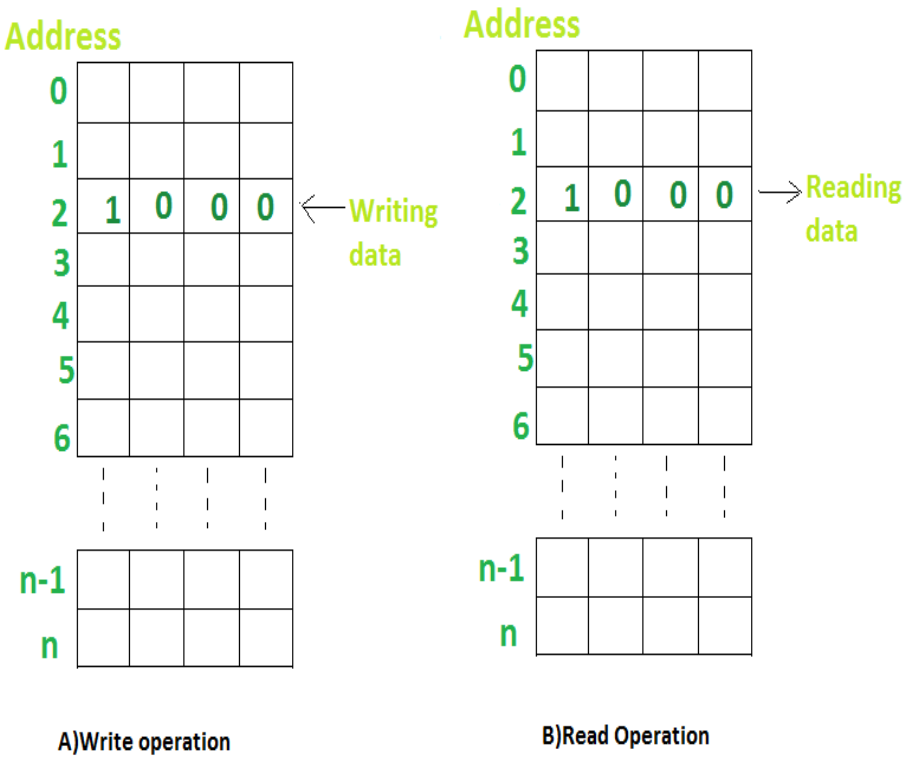
1. Introduction



The total memory of a computer can be visualized by hierarchy of components. The memory hierarchy system consists of all storage devices contained in a computer system from the ***slow*** Auxiliary Memory to fast Main Memory and to smaller Cache memory. The **main memory** occupies the central position because it is equipped to communicate directly with the CPU and with auxiliary memory devices through ***Input/output processor*** (I/O).

When the program not residing in main memory is needed by the CPU, they are brought in from auxiliary memory. Programs not currently needed in main memory are transferred into auxiliary memory to provide space in main memory for other programs that are currently in use.

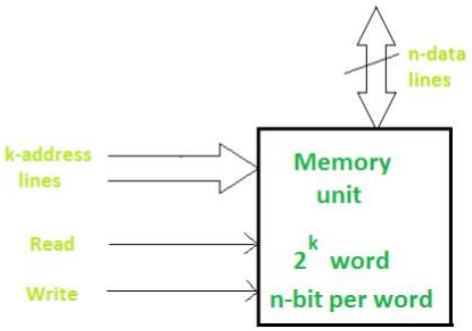
The main memory consists of a large number usually many thousands of storage cells each of which can store a binary digit or bit having the value 0 or 1 (**Cell** is a small storage element in memory). The data in memory are ***stored*** and ***retrieved*** by the process called ***writing*** and ***reading*** respectively.



A **word i**s a group of bits where a memory unit stores binary information. A word with group of 8 bits is called a byte.

Memory Locations and Addresses

A memory unit consists of data lines, address selection lines, and control lines that specify the direction of transfer. The block diagram of a memory unit is shown below**:**



Data lines provide the information to be stored in memory. The control inputs specify the direction transfer. The k-address lines specify the word chosen. When there are k address lines, 2k memory word can be .accessed

For example, a memory with 64k words and a word size of 1 byte, then this memory unit has 64 * 1024 = 65536 memory locations. The address of these locations varies from 0 to 65535.

Unit	No. of Bytes
Kilobyte (KB)	2^{10} (1024) bytes
Megabyte (MB)	2^{20} (1048576) bytes
Gegabyte (GB)	2^{30} (1073741924)
Terabyte (TB)	2^{40} bytes

Example-1: A computer has 32 MB of memory. How many bits are needed to address any single byte in memory?

Solution

The memory address space is 32 MB, or 225 (25×220). This means that we need log2 (225), or 25 bits, to address each byte.

Example-2: A computer has 128 MB of memory. Each word in this computer is eight bytes. How many bits are needed to address any single word in memory?

Solution

The memory address space is 128 MB, which means 27 x 210 = 227. However, each word is eight (23) bytes, which means that we have 224 words. This means that we need log2224, or 24 bits, to address each word.

Assignment of byte addresses

- Little Endian (e.g., in DEC, Intel)

low order byte stored at lowest address: byte0 byte1 byte2 byte3.

Eg: 46 (High Byte), 78, 96, and 54 (Low Byte): (32 bit data)

Address	Mem. Cont.
8000	54
8001	96
8002	78
8003	46

- Big Endian (e.g., in IBM, Motorolla, Sun, HP)

» high order byte stored at lowest address: byte3 byte2 byte1 byte0.

- In case of 16 bit data, aligned words begin at byte addresses of 0, 2, 4, …….
- In case of 32 bit data, aligned words begin at byte address of 0, 4, 8, …….
- In case of 64 bit data, aligned words begin at byte addresses of 0, 8, 16, ……..
- In some cases words can start at an arbitrary byte address also then, we say that word locations are unaligned.

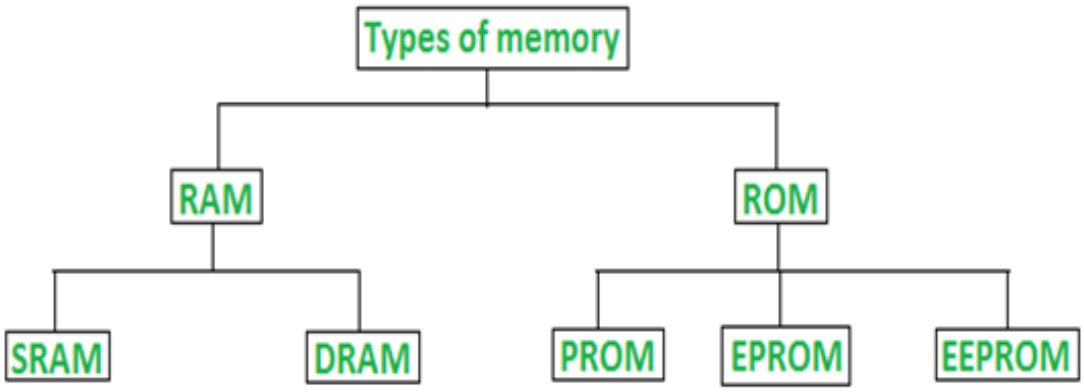
. Types of computer memory Memory

is the most essential element of a computing system because without it computer can't perform simple tasks. The memory of computer is broadly categories into two categories:

a) Primary (Internal): Internal memory is used by CPU to perform task.

b) Secondary (External): external memory is used to store bulk information, which includes large software and data.

a) **Primary memory Internal (**Primary) memories are semiconductor memory. Semiconductor memories are categorized as Random Access Memory (RAM) is volatile memory and Read Only Memory (ROM) is non-volatile memory



Classification of Primary Memory

1- Random Access Memory (RAM)

- It is also called as read write memory or the main memory or the primary memory.
- The programs and data that the CPU requires during execution of a program are stored in this memory.

- It is a volatile memory as the data loses when the power is turned off.
- RAM is further classified into two types-

a) Static-RAM (SRAM): which store the bit information in F/F.

b) Dynamic-RAM (DRAM: which store the bit information on a capacitor. So DRAM needs refreshing .

2- Read Only Memory (ROM):

- The information is stored permanently in such memories during manufacture.
- Stores instructions that are required to start a computer. This operation is referred to as bootstrap.
- It is not volatile.
- Always retains its data.
- Used in embedded systems or where the programming needs no change.
- Used in calculators and peripheral devices.
- ROM is further classified into 4 types- ROM, PROM, EPROM, and EEPROM.

Types of Read Only Memory (ROM)

1. **PROM (**Programmable read-only memory): It can be programmed by user. Once programmed, the data and instructions in it cannot be changed.

2. **EPROM** (Erasable Programmable read only memory): It can be reprogrammed. To erase data from it, expose it to ultra violet light. To reprogram it, erase all the previous data .

3. **EEPROM** (Electrically erasable programmable read only memory): The data can be erased by applying electric field, no need of ultra violet light. We can erase only portions of the chip .

RAM	ROM
1. Temporary Storage.	1. Permanent storage.
2. Store data in MBs.	2. Store data in GBs.
3. Volatile.	3. Non-volatile.
4.Used in normal operations.	4. Used for startup process of computer.
5. Writing data is faster.	5. Writing data is slower.

Difference between RAM and ROM

b) Secondary (External) memory This type of memory is also known as external memory or non-volatile. It is slower than the main memory (RAM). These are used for storing data/information permanently. CPU directly does not access these memories; instead they are accessed via input-output routines. The contents of secondary memories are first transferred to the main memory, and then the CPU can access it. For example, disk, CD-ROM, DVD, etc.

4. Main Memory Organization

The number of bits stored in a register is called a memory word. Memory devices (chips) are available in variable word size. Each word in memory has specific binary number called memory address .

To communicate with the main memory, the CPU should be able to :

- Select the chip.
- Identify the memory location, and
- Identify the memory operation (Read from / Write into) the memory location .

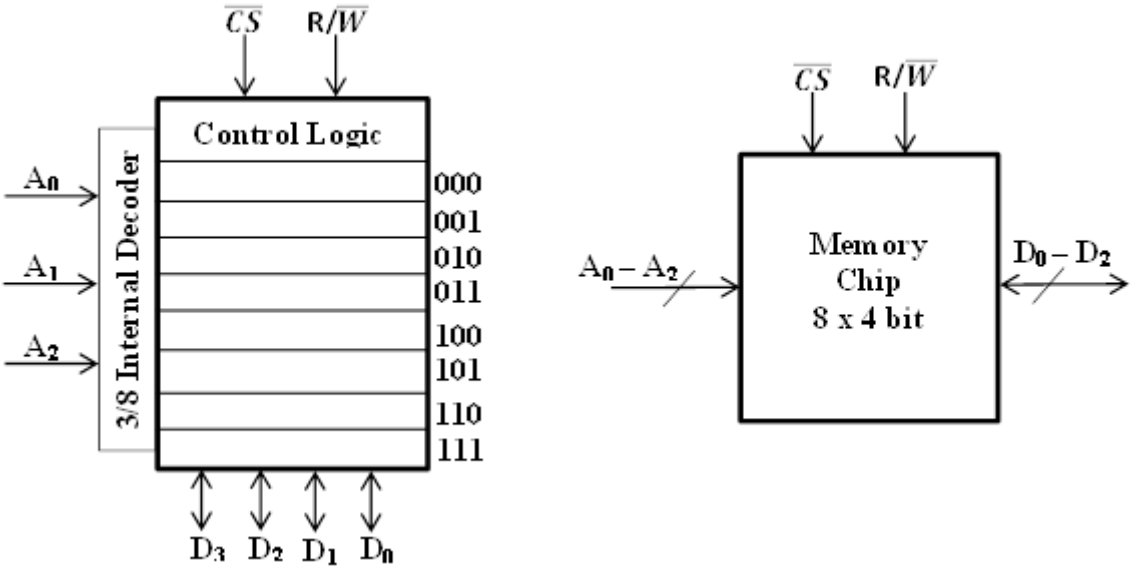
To do its job, any memory chip must have the following terminals:

a) **Address lines**: to identify number of memory locations in the chip according to the following relation:
n-address line ➔ 2^m memory locations
e.g : if n=9 then the chip contains 2^9 = 512 locations .

b) **Data lines**: represent data input / output lines in a memory chip.
e.g: 2M x 8 of memory has 8-data lines and 21 address lines since,
2M= 2 x 2^{20} = 2^{21}

c) **Control lines**: memory chips are provided two control lines, $\overline{R/W}$ (Read or Write Line) and $\overline{CS}$ (Chips Select). The $\overline{R/W}$ line is used to specify the required operation about read or write. Chip Select line is required to select a given chip in a multi- chip memory system .

The following figure shows hypothetical memory chip of eight registers (locations), 3 address lines, one chip select $\overline{CS}$, one $\overline{R/W}$ line and 4 data lines.

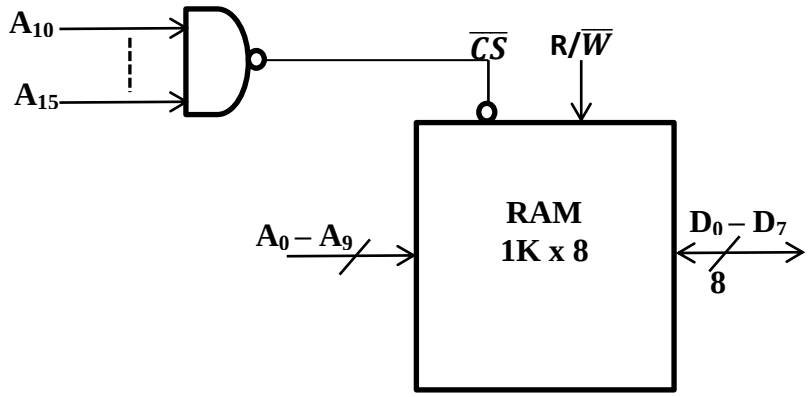


Note: The address bus lines are split into two sections: N most significant bits for CS and M least significant bits addressing memory locations.

e.g: How many memory locations can be addressed by a CPU with 16 address lines?

Solution: $2^{16} = 2^6 \times 2^{10} = 64K$.

e.g : Give the effective address for the following memory chip?



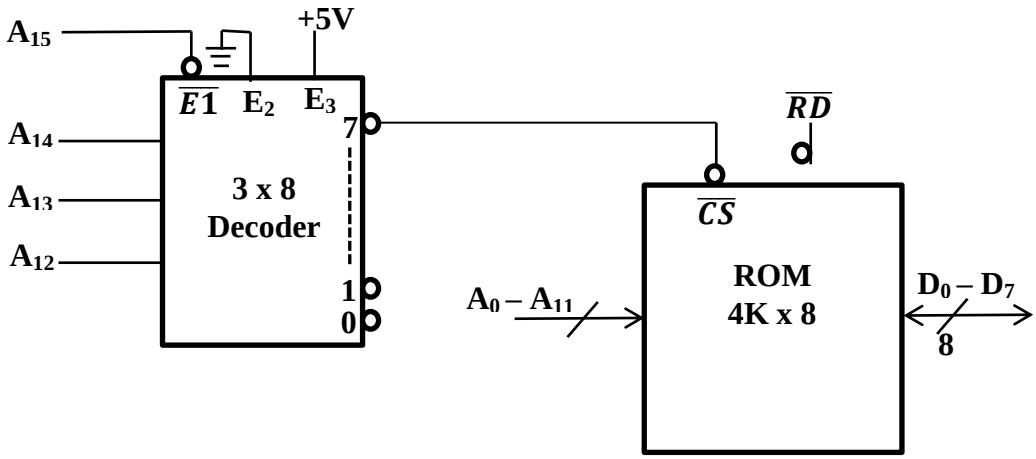
Solution:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	= $F800_h$

1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 = $FBFF_h$

Chip select Location select

e.g: Give the effective address for the following memory chip?



Solution:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	= 7000_h Start
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	= $7FFF_h$ End

2.2 Main Memory Expansion

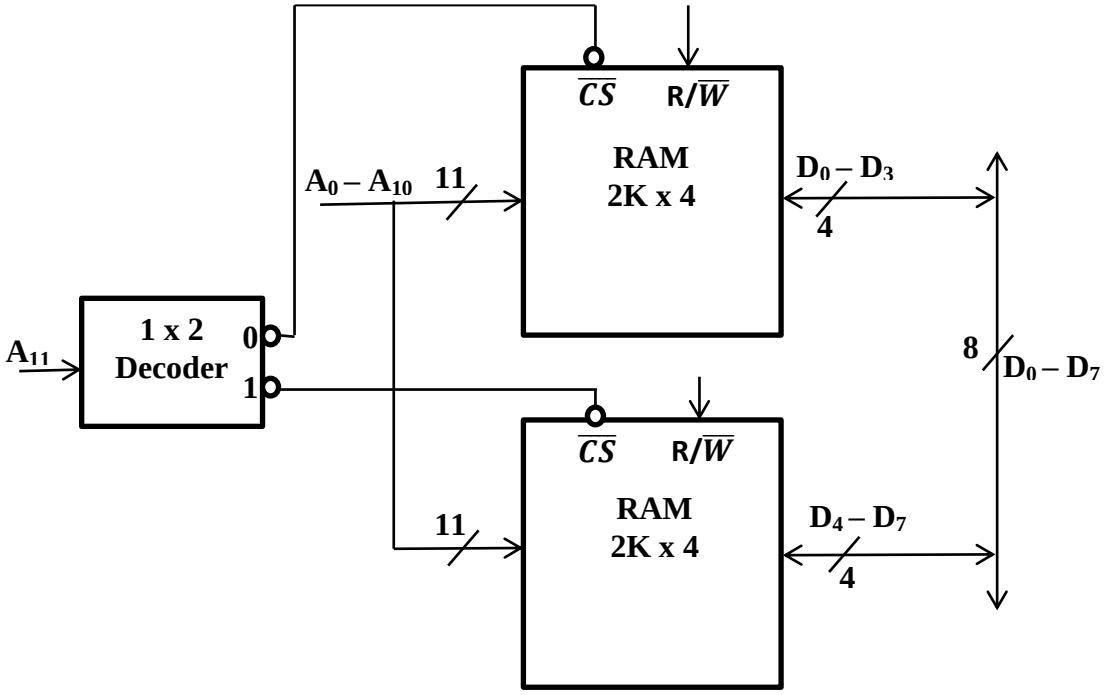
A) Horizontal Expansion (increase number of data)

e.g :- Implement/ Design a 2Kbytes memory using 2Kx4bit chips RAM in CPU has 14 address line and 8 bit data bus?

$$No.of\ chips = \frac{Total\ size}{size\ of\ a\ chip} = \frac{2K \times 8}{2K \times 4} = 2\ chips$$

$$no.of\ address\ lines\ for\ each\ chip: 2K = 2 \times 2^{10} = 2^{11}$$

$$\Longrightarrow 11\ address\ lines\ common\ between\ the\ chips\ (A_0 - A_{10})$$



A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0		
0	0	0	0	0	0	0	0	0	0	0	0	0	Start — **RAM1**
0	1	1	1	1	1	1	1	1	1	1	1	1	End — **RAM1**
1	0	0	0	0	0	0	0	0	0	0	0	0	Start — **RAM2**
1	1	1	1	1	1	1	1	1	1	1	1	1	End — **RAM2**

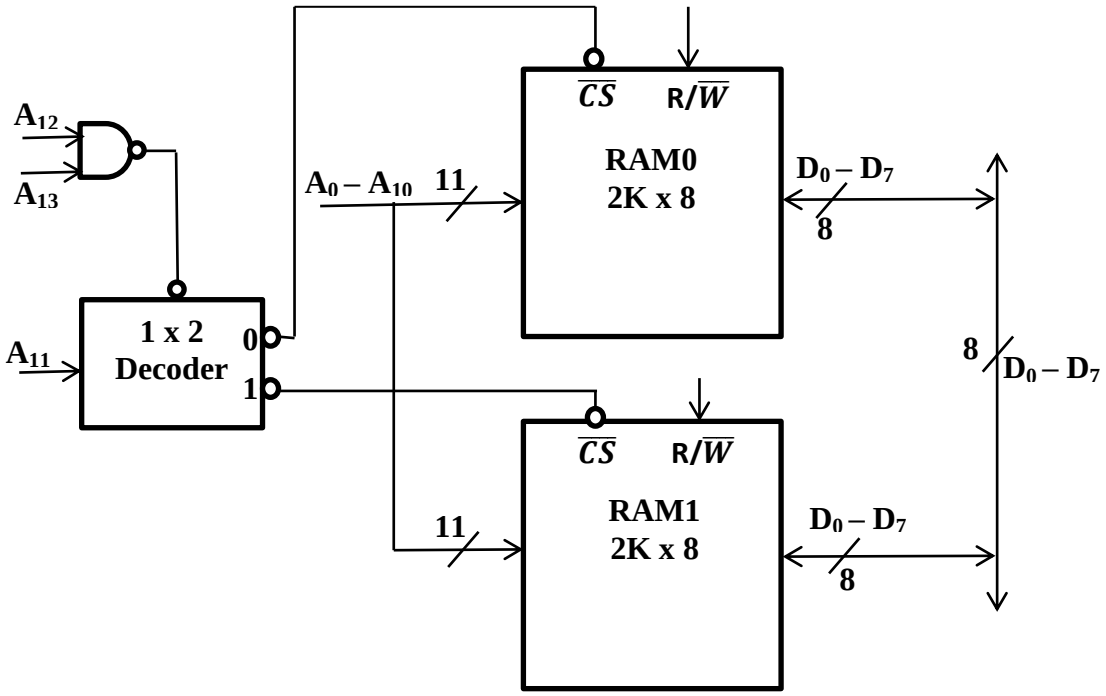
B) Vertical Expansion (increase number of address line)

e.g:- Implement a memory with 4KB using 2KB RAM chips. If the CPU has 14 address line?

$$No.of\ chips = \frac{Total\ size}{size\ of\ a\ chip} = \frac{4K\ \times 8}{2K \times 4} = 2\ chips$$

$$no.of\ address\ lines\ for\ each\ chip: 2K = 2 \times 2^{10} = 2^{11}$$

$$\Longrightarrow 11\ address\ lines\ common\ between\ the\ chips\ (A_0 - A_{10})$$



3. Instruction Format

The bits of the instruction are divided into groups called fields

1- An operation code field (op-code): is a group of bits that define various processor operations such as ADD, Complement and shift.

2- An address field: that designates a memory address or processor register.

3- A mode field: that specifies the way the operand or the effective address is determine.

Op-Code	Mode	Address

4. Addressing Mode :

The computer use the addressing mode :

- To give different type of memory addressing
- To reduce the number of bits in the address field of the instruction addressing modes

1- **Implied Mode :** In this mode the operands are specified implicitly in the definition of the instruction.

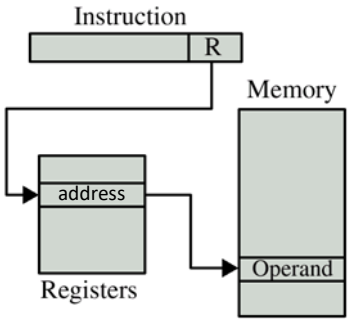
2- **Immediate Mode :** Operands is specified in the instruction itself

Ex:- instruction for initializing registers to a constant value.

3- **Register Mode :** The operands are in register that reside to a constant value.

4- **Register Indirect:** The instruction a register in the CPU whose contents give the address of the operand in memory.

Effective address (EA) = [processor register]



5- **Auto Increment & auto decrement Mode:** The same operation of register indirect mode but with increment or decrement by one.

6- **Direct addressing Mode:** Effective address (EA)= [address field in the instruction]

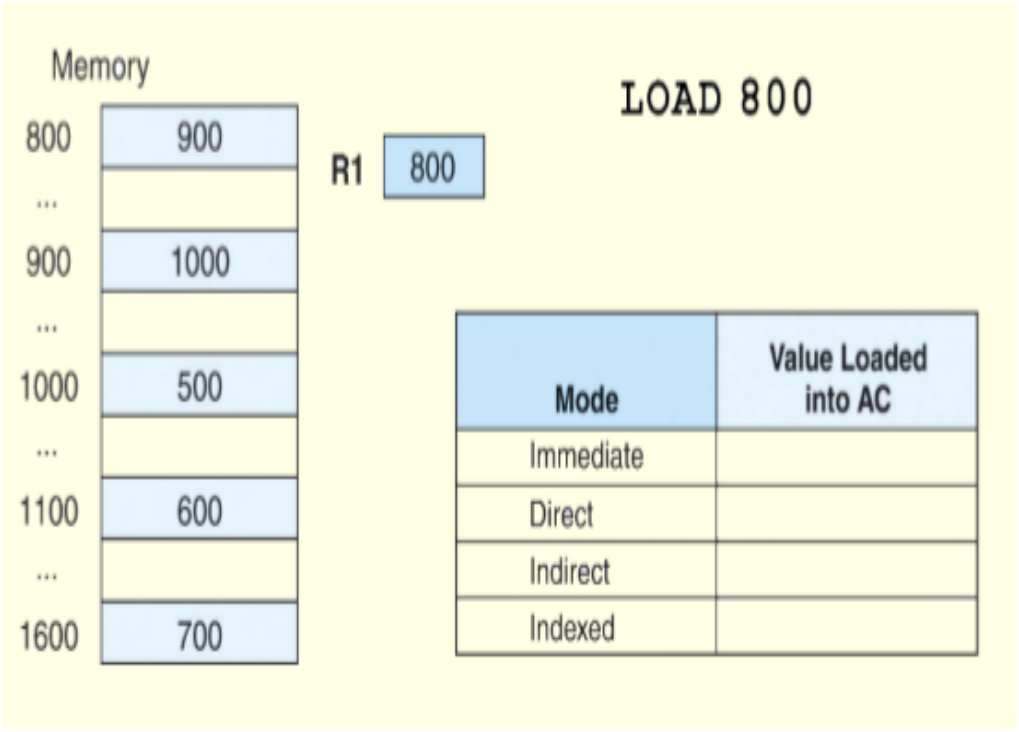
7- **Indirect Addressing Mode :** Effective address (EA) = M [address field]

8- **Relative Address Mode :** Effective address(EA) = [PC] + address field (displacement)

9- **Based Addressing Mode :** Effective address(EA) = [BR] + address field (displacement)

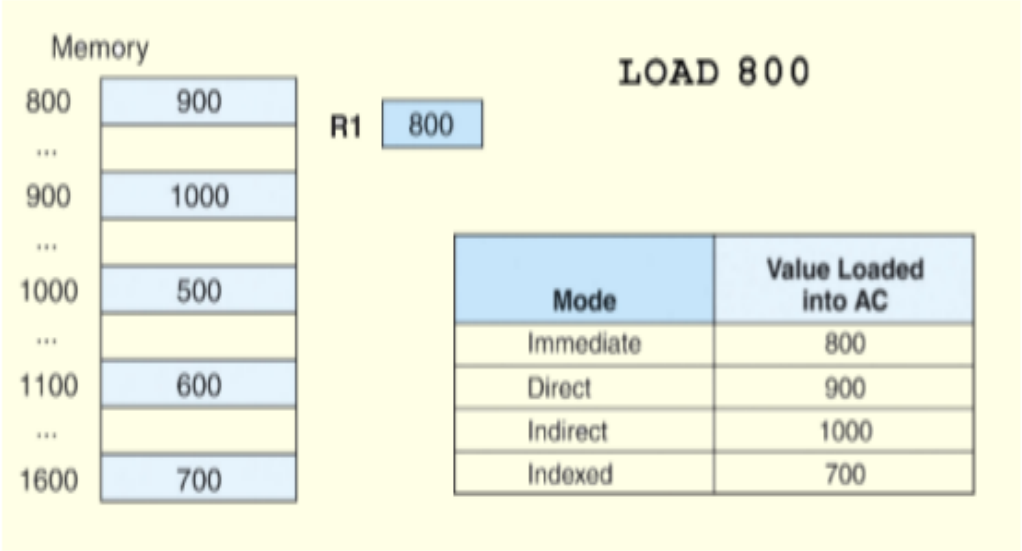
10- Indexed Addressing Mode : Effective address (EA)= [XR] or [IR] + address field (displacement)

Ex: For the instruction shown, what value is loaded into the accumulator for each addressing mode?



Mode	Value Loaded into AC
Immediate	
Direct	
Indirect	
Indexed	

These are the values loaded into the accumulator for each addressing mode.



Mode	Value Loaded into AC
Immediate	800
Direct	900
Indirect	1000
Indexed	700

e.g: Consider a memory contents shown in figure below. The two word instruction at address 200 & 201 is "Load to Acc." Instruction with an address field equal to 500?

PC = 200

R1 = 400

XR = 100

Acc

200	Load to Acc	Mode
201	Address= 500	
202	Next instruction	
399	450	
400	700	
500	800	
600	900	
702	325	
800	300	

Solution:

Addressing Mode	Effective Address	Content of Acc
Direct		
Immediate		
Indirect		
Relative		
Indexed		
Register		
Register Indirect		
Auto Increment		
Auto Decrement		

Table below shown the recommended assembly language conversion & the actual transfer accomplished in each case where

ADR strand for an address

NBR is a number or operand

X is an index register

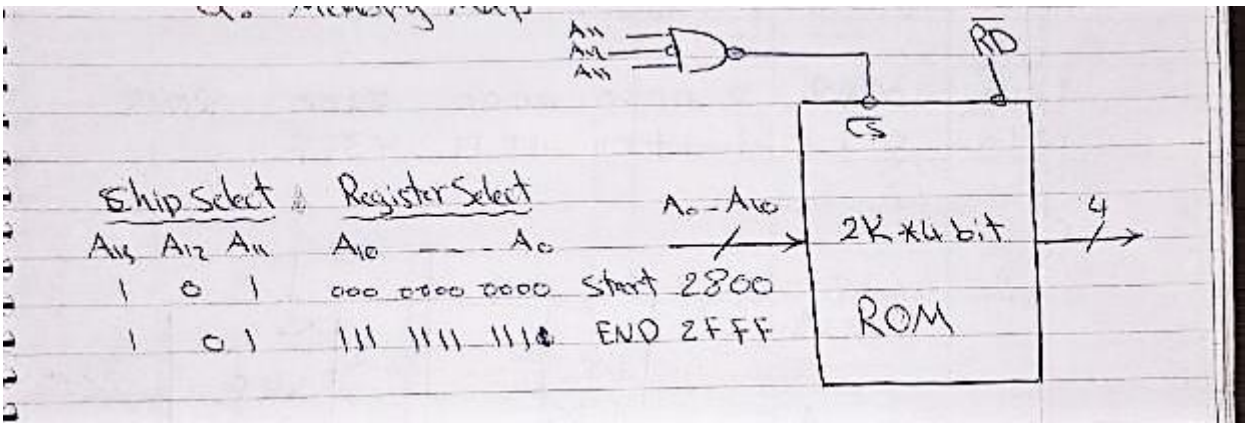
R1 is processor register

Ac is accumulator

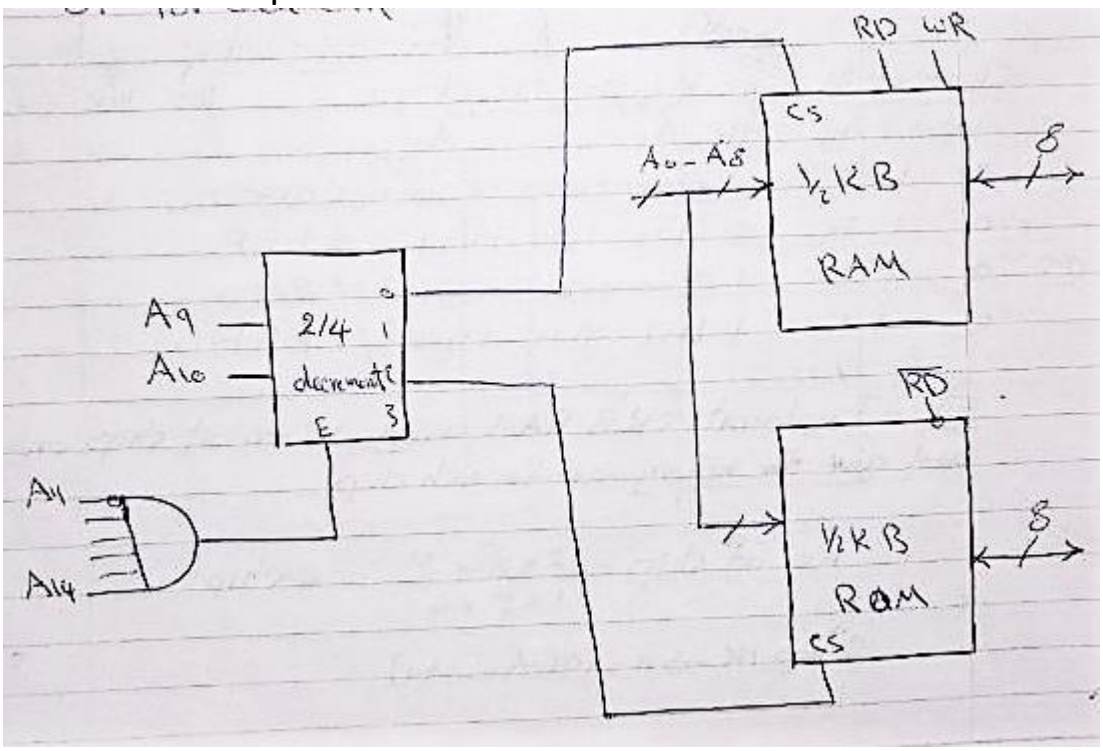
Mode	Assembly Conversion	Register Transfer
Direct	LD ADR	AC ← M [ADR]
Indirect	LD @ADR	AC ← M [M [ADR]]
Relative	LD $ADR	AC ← M [PC + ADR]
Immediate	LD #NBR	AC ← NBR
Index	LD ADR(X)	AC ← M [ADR + XR]
Register	LD R1	AC ← R1
Register Indirect	LD (R1)	AC ← M [R1]
Auto Increment	LD (R1)+	AC ← M [R1] , R1← R1+1
Auto Decrement	LD –(R1)	AC ← M [R1-1] , R1← R1-1
Index Indirect	LD @ADR(X)	AC← M [X+M[R1]]

e.g : Give the effective address for the following.

a. Memory Map



b. For each chip



Chip Select

A14 A15 A10 A9

1110 0 0

1110 0 1

Register Select

A8 ——— A0

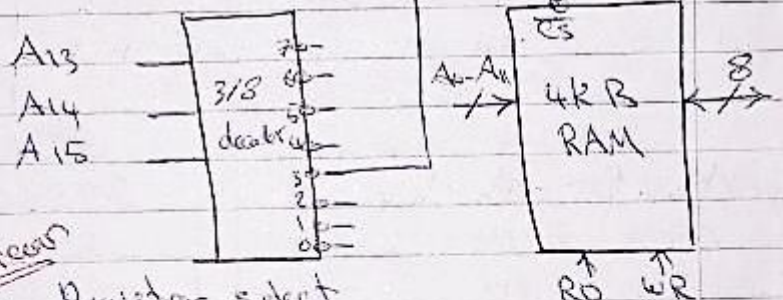
0 0000 0000 7000 RAM

1 1111 1111 71FF

0 0000 0000 7200 ROM

1 1111 1111 73FF

c.



Chip Select

A15 A14 A13

0 1 1

0 1 1

0 1 1

0 1 1

Register Select

A12 A11 ——— A0

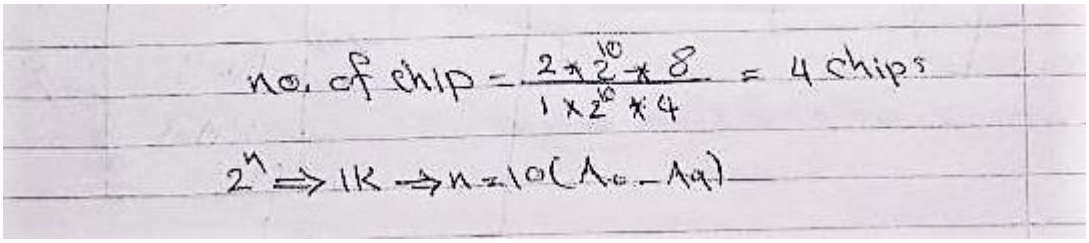
0 0000 0000 0000 6000

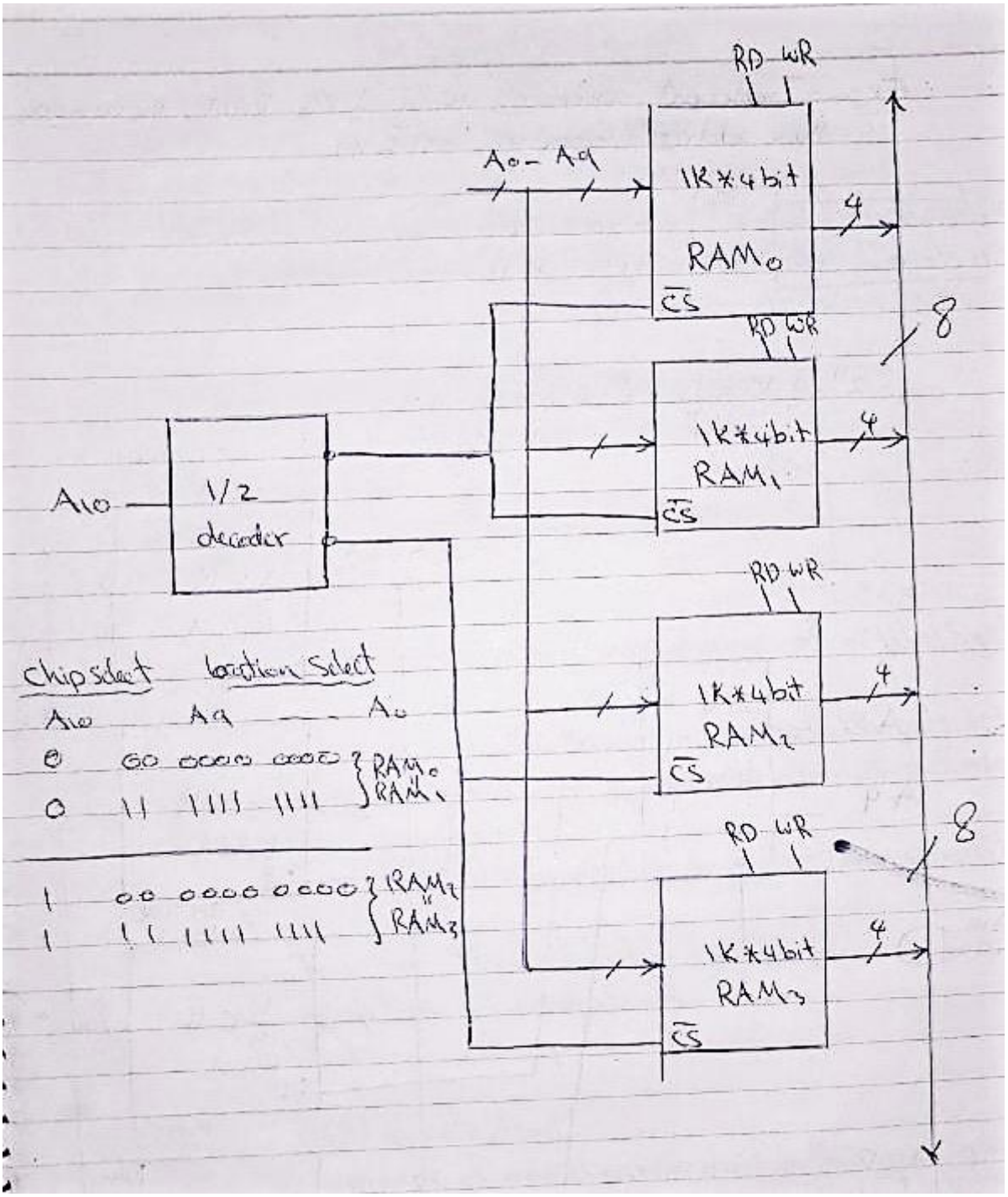
0 1111 1111 1111 6FFF

1 0000 0000 0000 7000

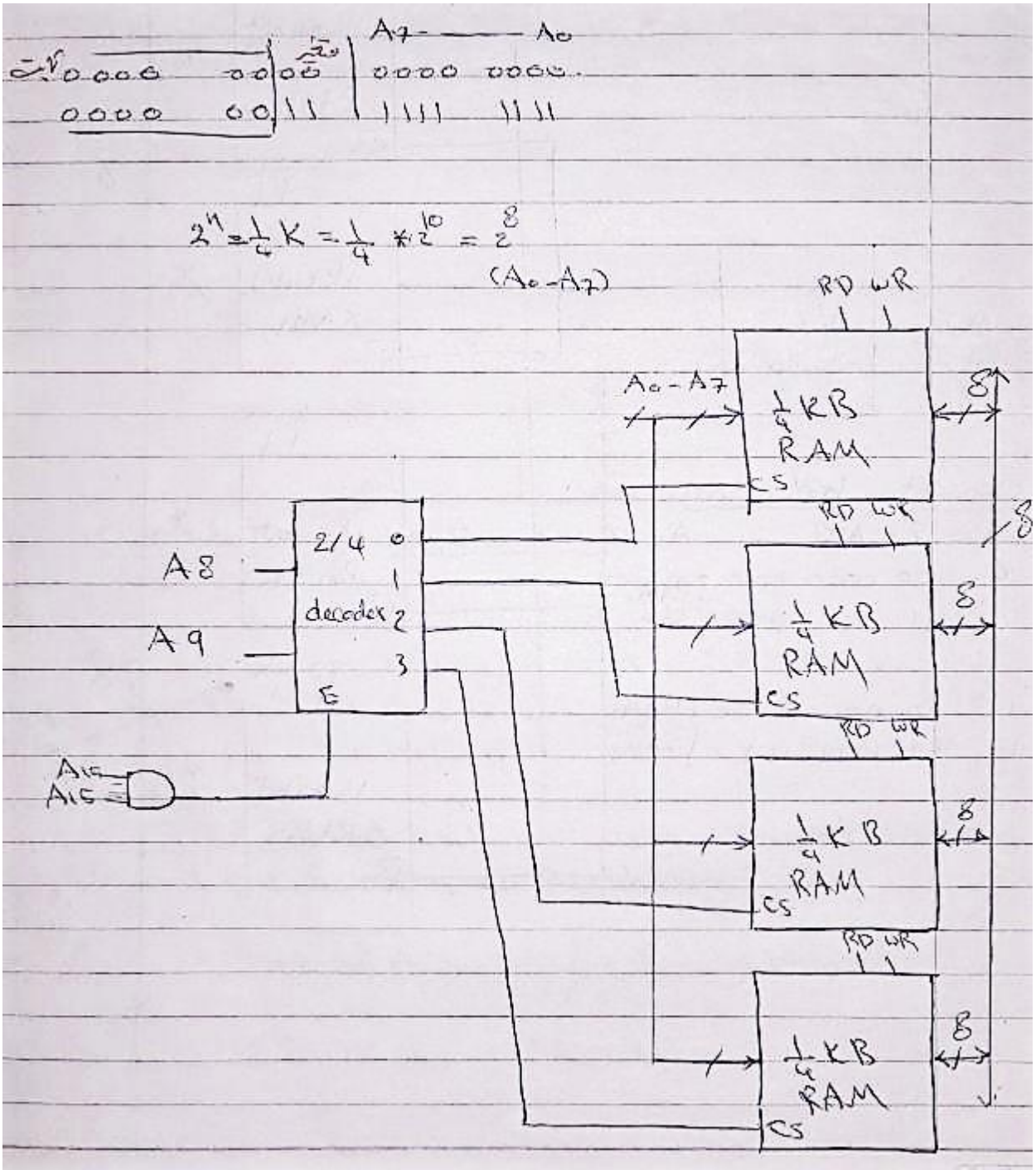
1 1111 1111 1111 7FFF

e.g : Implement 2KB RAM using 1Kx4 bit chip only and give the memory map for each chip.





e.g : Implement a memory using 1/4KB RAM, the memory have address (0000_H – $03FF_H$)



Example2: Main Memory = 1M × 8bits, RAM chips = 256K × 4bit How many RAM Chips require ?

Solution

For this memory we require 4 × 2 = 8 RAM chips.

Each chip requires 18 address bits (ie. 218 = 256K).

And 1M × 8 bits requires 20 address bits (ie. 220 = 1M)

H.W/An array of two integers (an integer is 32 bits wide) is placed in memory starting with address 100. Show how to increment the elements of the array using displacement addressing. The memory is byte addressable.