Computer Organization & Architecture

Unit - 4

Memory hierarchy- memory technologies- cache memory- mapping & improving cache performance

* Memory Hierarchy

→ consists of multiple levels of memory with different speeds & sizes

→ Faster memories are smaller and more expensive per bit

Speed	Processor	Size	Cost (/bit)	Current Technology
Fastest	Memory	Smallest	highest	SRAM
	Memory			DRAM
Slowest	Memory	biggest	Lowest	magnetic disk

→ At the top of the hierarchy are registers that are matched in speed to the CPU.

→ There are a small no. of registers in a processor, of the order of a few hundred or less.

→ At the bottom of the hierarchy are the secondary memories such as magnetic disks / tapes where the cost per stored bit is small in terms of money and electrical power, but the access time is very long compared to registers

→ As we move up the hierarchy, greater performance is realized, at a greater cost.

* Terminology

1. **Block** : The minimum unit of information that can be either present or not present in the two-level hierarchy is called a block or a line.

2. **Hit** : If the data requested by the processor appears in some block in the upper level, it is called a hit

3. **Miss** : If the data is not found in the upper level, the request is called a miss. The lower level in the hierarchy is then accessed to retrieve the block containing the requested data.

4. **Hit rate / hit ratio**: fraction of memory accesses found in the upper level, used to measure the performance of the memory hierarchy

5. **Miss rate**: 1 - hit rate = fraction of memory accesses not found in the upper level

6. **Hit Time** : Time to access the upper level of the memory hierarchy, which includes the time needed to determine whether the access is a hit or a miss.

7. **Miss Penalty** : Time to replace a block in the upper level with the corresponding block from the lower level, plus the time to deliver this to the processor

* Principle of Locality

→ states that programs access a relatively small portion of their address space at any instant of time. There are 2 types of locality:

(a) Temporal Locality - The principle stating that if a data location is referenced, it is likely to be referenced again soon

(b) Spatial Locality - The locality principle stating that if a data location is referenced, data locations with nearby addresses will likely be referenced soon.

* Memory Technology

There are 4 primary technologies used today in memory hierarchies:

(i) DRAM (Dynamic RAM) ⇒ for main memory
(ii) SRAM (Static RAM) ⇒ for caches
(iii) Flash Memory ⇒ non-volatile, secondary memory in mobile device
(iv) Magnetic Disk ⇒ largest & slowest

A. SRAM Technology

→ are integrated circuits that are memory arrays, with a single access port that can provide either a read or write
→ SRAMs have fixed access times to any data.
→ do not need to refresh ⇒ access time is close to cycle time
→ as long as there is power, the value can be indefinitely retained.

B. DRAM Technology

→ The value kept in a cell is stored as a charge in a capacitor, a single transistor is used to access this stored charge (hence, capacitor)

→ As DRAMs store the charge on a capacitor, it cannot be kept indefinitely, and must periodically be refreshed. (hence called dynamic)

→ Refreshing ⇒ reading the contents & writing it back

→ Some DRAMs have clocks to synchronize w/ the processor - called Synchronous DRAMs / SDRAMs.

adv = eleiminates time for memory & processor to synchronize

→ Also a version called Double Data Rate (DDR) SDRAM ⇒ data transfers on both the rising & falling edge of the clock, meaning there is twice the bandwidth.

C. Flash Memory

→ a type of electrically erasable programmable read only memory (EEPROM)

→ used in mobile devices

→ wear leveling : spread writes by remapping blocks that have been written many times to less trodden blocks.

D. Disk Memory

→ a collection of platters that rotate on a spindle.

→ have magnetic recording material on both sides.

→ has a movable arm w/ a small electromagnetic coil

→ tracks - disk surface divided into concentric circles

→ sector : track divided into sectors that contain the information

Seek Time: time taken to move head to desired track

Rotational latency: time taken for the desired sector to rotate under the read / write head, at the correct track. called rotational latency.

*Cache Memory

→ represent the level of memory hierarchy between the processor and the main memory

→ The fundamental idea of cache organization is that by keeping the most frequently accessed instruction and data in the fast cache memory, the average memory access time will be reduced.

→ Typical mapping between address and cache locations for a cache is computed as:

> block = block address % no. of blocks in the cache

→ If the no. of entries in the cache is a power of 2, then the modulo can be computed by using the low-order $\log_2$(cache size in blocks)

*Accessing a cache

→ Each cache location can contain the contents of a number of different memory locations

→ A tag is used to identify whether the word in the cache corresponds to the requested word.

→ The tag needs only to have the upper portion of the address, corresponding to the bits that are not used to index the cache (The lower 3 bit index field of the address selects the block)

→ A valid bit is used in the tables to indicate that the associated block in the hierarchy contains valid data.

Calculating Tag Size & Cache Size

If cache block size = k

block size = 2^m words

$2^n = k$

m bits for each word

n = no. of bits for the index

if the address length is A (say 32)

the no. of bits in the tag is $A - (n+m+2)$

More Formulae

* Cache Mapping Techniques

Direct Mapping

→ A cache structure in which each memory location is mapped to exactly one location in the cache.

→ calculated as:

(Block address) modulo (no. of blocks in cache)

→ Contention may occur:

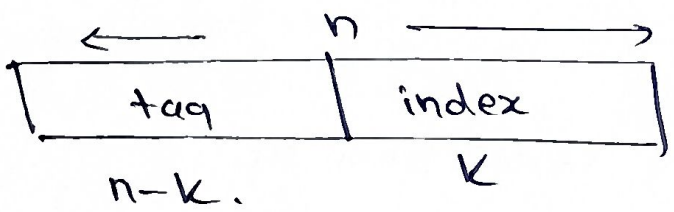
(i) when the cache is full

(ii) when more than one memory block is mapped to a given cache block position.

Solution : new blocks override the current resident

In general : 2^k words in cache memory

2^n words in main memory



∴ Each word in the cache consists of the data word & the associated tag.

Advantages → easy to implement

Disadvantages → not very flexible

→ hit ratio can drop considerably if 2 or more words whose address has the same index, but diff. tags are repeatedly accessed.

* Fully Associative

→ a block can be placed in any location in the cache

→ To find a given block in a fully associative cache, all entries have to be searched because a block can be placed in any one.

Advantage → more flexible than direct mapping

Disadvantage → increases hardware cost

→ only suitable for caches w/ a small no. of blocks

* Set Associative

→ there are a fixed no. of locations where each block can be placed.

→ A set-associated cache with n locations for a block is called n-way set associative.

→ An n-way set associative cache consists of a no. of sets each of which has n-blocks.

A set containing a memory block is given by

(Block no) modulo (no. of sets in cache)

Numericals

(1) 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 four words

a. How many bits are there in the tag, index, block & word fields of the address?

b. How many bits are there in each word of the cache, and how are they divided into fns? Include a valid bit

c. How many blocks can be accommodated?

Main memory = 64k x16

= 64k words of 16 bits each

Main memory size = 64k words

$= 2^6 \times 2^{10}$

$= 2^{16}$ words

Address = 16 bits

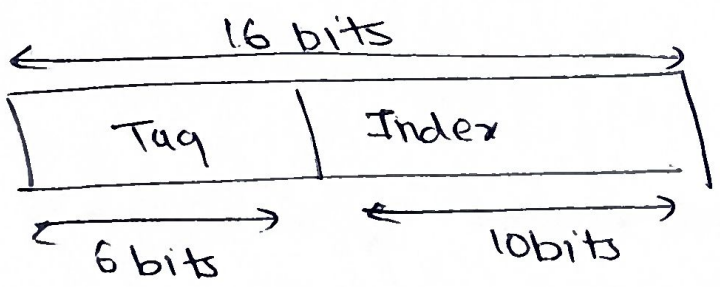
word size = 16 bits each

Data = 16 bits

Cache memory = 1K (length of data = 16 bits, due to direct mapping)

$= 2^{10}$ words

Index = 10 bits



No. of bits in a block = ?

Total no. of blocks = 1K = 1024

size of a block = 4 words

No. of blocks = $\frac{1024}{4} = 256 = 2^8$

no. of bits in a block = 8

No of words in a block = 4

$= 2^2$

no. of bits to access word = 2

b. Total size

= valid bit + tag + address

= 1 + 6 + 16

= 23 bits in each word of cache memory

c. cache can accomodate 256 blocks.

(2) A two-way set associative cache memory uses blocks of 4 words. The cache can accommodate a total of 2048 words from the main memory. The main memory size is 128K x 32

a. Formulate all the pertinent information required to construct the cache memory. (Tag, index, data, blocks, words)

b. What is the size of the cache memory?

Ans: Main memory:-

128K x 32 = 128K words of 32 bits each

address = 128K

$= 2^7 \times 2^{10}$

$= 2^{17}$

address = 17 bits

data = 32 bits

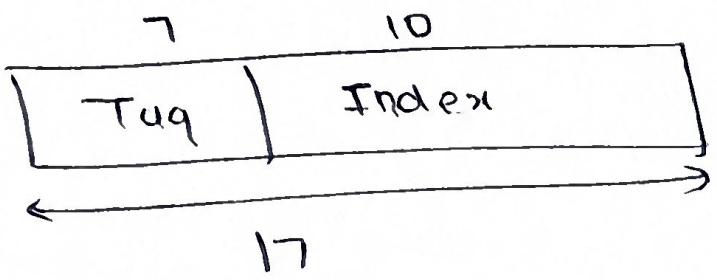
Cache memory: 2048 words

2-way set associative

= 1024 words

$= 2^{10}$ words

index = 10 bits ~~words~~



Tag = 7 bits

bits required to represent:

a word = 2^2 = 2 bits

no. of blocks = $\frac{1024 \text{ words}}{4 \text{ words}}$

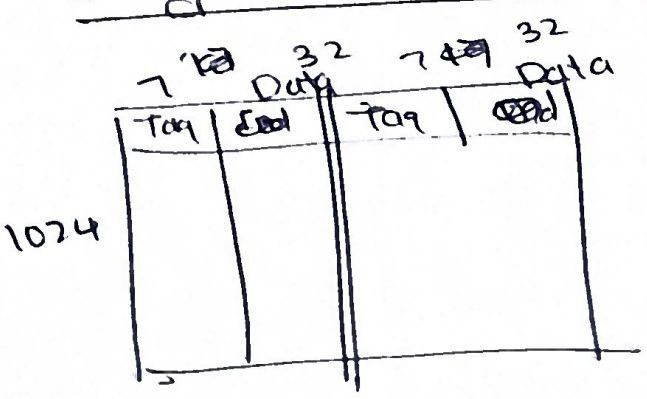
= 256 blocks

block + word = index

$= 2^8$

block = 8 bits

Size of the cache memory



to find total cache size

use tag & data

= 1024 × 78

= 1K × 78

cache memory size = 78K

(3) How many 128 x 8 RAM chips are needed to provide a memory capacity of 2048 bytes?

a 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?

b How many lines must be decoded for the chip select? Specify the size of the decoder?

1 byte = 8 bits

Main memory = 128 x 8 RAM

address = 128 = 2^7

address = 7 bits

data = 8 bits

Cache memory = 2048 bytes

= 2^{11}

index = 11 bits

| block | word |

| tag | index |

11

(a) given 128 x 8 bits

required mem. capacity = 2048

no. of chips = $\frac{2048}{128}$ = 16 chips

(b) To access 2048 bytes : 2^{11} ⇒ 11 address lines

no. of chip bits to distinguish among 16 bits = 4

common lines = 11 − 4 = 7

need a 4 × 16 decoder

(4) Check

A computer uses RAM chips of 1024 × 1 → bit capacity.

(a) How many chips are needed to provide a memory capacity of 1024 bytes

(b) How many chips are needed to provide a memory capacity of 16kB.

assume that it is 1024 × 8

Ans: 1024 × 8

(5) A direct mapped cache has a capacity of 16kB and a line length of 32 bytes. How many bits are used to determine the byte that a memory op. referenced within a cache line, how many bits are used to select the line in the cache that may contain the data.

cache size = 16kB

$= 2^4 \times 2^{10} = 2^{14}$

= 14 bits

line length = 32 bytes

$\therefore \frac{2^{14}}{2^5} = 2^9$ ⇒ 9 bits to select line

word offset = $\log_2 32 = 5$

⇒ 5 bits to determine which byte is being referenced

(6) If a cache has 64 byte cache lines, how long does it take to fetch a catchline, if the main memory takes 20 cycles to respond to each memory request & return 8 bytes of data in response to each request?

Ans no of memory requests needed = 64/8 = 32

no of cycles per req = 20

Total no of cycles = 640

(7) An address space is specified by the 24 bits & the corresponding memory space by 16 bits

(a) How many words in address & mem. space

(b) If the page has 2k words, how many pages & blocks are there in the system?

Ans:

Words in address space = 2^{24} = $2^{20} \times 2$ 16M words

memory space = 2^{16} = 64k words

pages = 16/2k = 8k pages

blocks = 64k / 2k = 32 blocks

(8) The logical address space in a computer system consists of 128 segments Each segment has upto 32 pages of 4k words each. Physical memory has 4k blocks of 4words each. Find logical & physical addr. formats

Logical Address

7	5	12
seg	page	word

Physical

12	10
Block	word

Computer Organization and Architecture

Unit 4

Additional Numericals

(1) Given ROM size of 512 kB, estimate the no. of address lines and data lines needed to access the memory

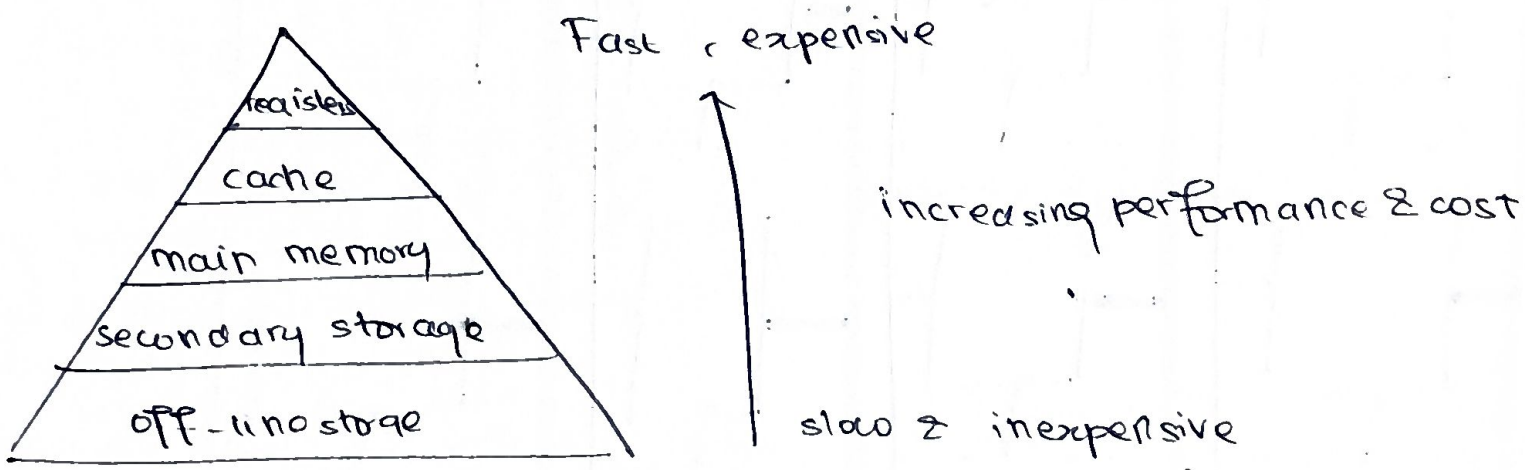
$512\ kB = 2^9 \times 2^{10}$

$= 2^{19}$

= 19 bits for address line

8 bits for data line

(2) Show structure of memory hierarchy



(3) Construct a static RAM memory of size 2M x 32 using 512 K x 8 static memory

RAM memory: $2M \times 32$

$= 2 \times 2^{20} \times 2^5 = 2^{26}$

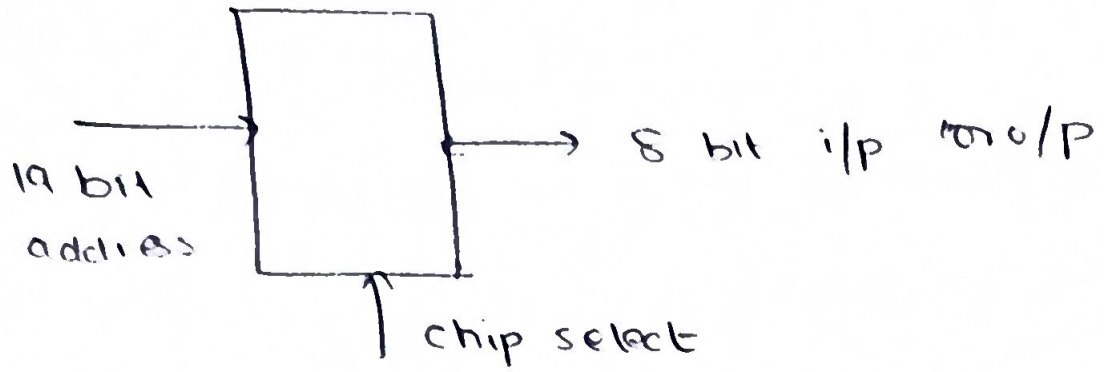
static memory: $512K \times 8$

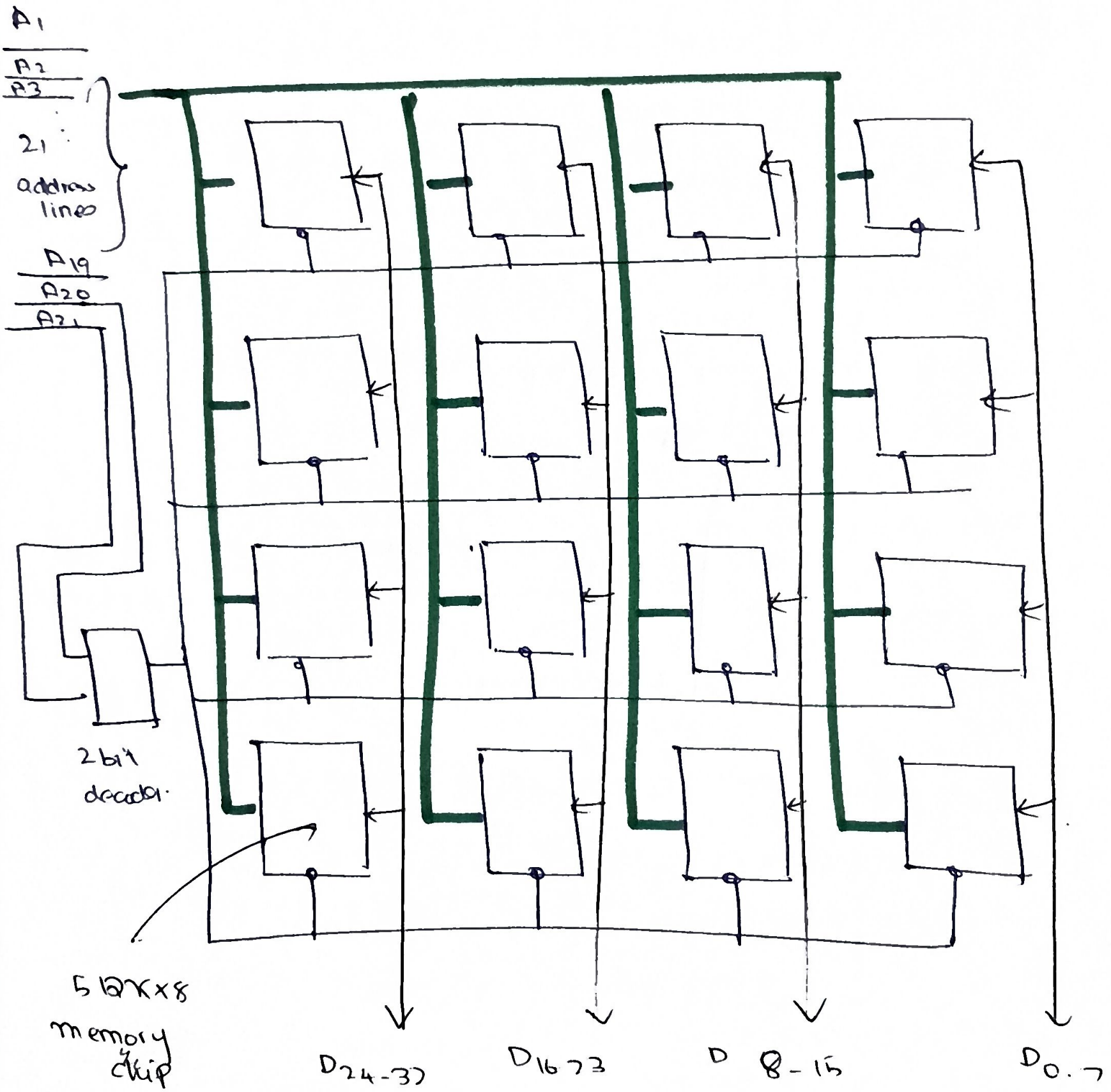
$= 2^9 \times 2^{10} \times 2^3 = 2^{22}$

no. of blocks = $\frac{2^{26}}{2^{22}} = 2^4$

= 16 blocks

512 K x 8 memory chip





Computer Organization and Architecture

Unit 4

* Measuring and Improving Cache Performance

Measurement: done using hit and miss rates

CPU Time = (CPU execution clock cycles + memory stall cycles) × clock cycle time

memory stall cycles: from cache misses

Memory stall cycles can also be defined as: read-stall cycles + write-stall cycles

Improvement: **reduce the miss rate** → reduce the probability that 2 different memory blocks will contend for the same cache location

reduce the miss penalty → add an additional level to the hierarchy, called multilevel caching

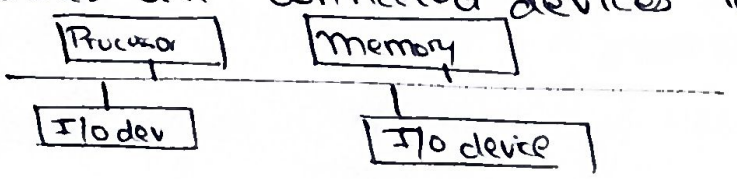
* Virtual memory + TLB – refer OS notes

* Accessing I/O Devices

→ done with a bus

↳ a shared communication link, which uses one set of wires to connect multiple subsystems.

→ enables all connected devices to share information



→ Buses consist of 3 sets of lines

(a) Address line : processor places a unique address to an I/O device on address lines

(b) Data: data placed on data lines, carry info between src & dest

(c) Control: used to signal requests and acknowledgement, and to indicate what kind of info is on the data lines

* Input - Output System

is of 2 types:

(i) memory - mapped I/O

(ii) programmed I/O

A. Memory - Mapped I/O

→ I/O devices and memory share the same address space

→ portions of address space are assigned to I/O device

→ reads and writes to those portions are interpreted as commands to the I/O

B. Programmod I/O

→ requires all I/O operations to be executed under the direct control of the CPU.

→ data transfer happens between a registers - one is the CPU register and the other is attached to the I/O device. I/O devices does not have access to main memory.

→ CPU executes several instructions to transfer data from I/O to memory

* Interrupts

→ causes a CPU to temporarily transfer control from its current program to another program

→ An interrupt signal is sent via a dedicated ~~process~~ line to the processor

Processor executes the interrupt service routine (ISR)

→ All registers, flags, PC values are stored onto a stack

→ Time required to save status & restore contributes to Interrupt Latency

hardware interrupt - power supply failure

software interrupt - division by 0

Steps

Assert interrupt request line

↓

Complete current instruction

↓

Put current PC value onto stack

↓

Load PC value w/ addr. of ISR

↓

Execute return from interrupt

↓

Continue where interrupted

* Handling Interrupts

1. Maskable Interrupt - can be masked under software control

2 Non maskable Interrupt - cannot be masked under software control

3. Single line Interrupt - all I/O ports share a single interrupt request line

   has a polling software that checks the load state of each device

4. Multilevel Interrupt - I/O device has individual interrupt pins

   no polling required - save time

5. Vectored Interrupts - Each device assigned an interrupt vector.

   Interrupts are handled by the processor after the device acknowledges w/ the interrupt vector as id

6. Interrupt Nesting - preemption of low priority interrupt by another high priority interrupt

7. Daisy Chaining - INTR common to all devices
   (int. req. line)

   INTA (ack. line) propagates in a serial manner

   closest device (serially) is of higher priority

8. Daisy Chaining w/ a Priority Group - combine daisy chaining and interrupt nesting to form priority group.

   Each priority group has daisy chaining within.

9. Pipelined Interrupts - difficult to find interrupting instruction in pipelined, precise vs. imprecise interrupts

* Direct Memory Access

→ used when large blocks of data have to be transferred at high speeds

→ DMA allows transfer directly between I/O and memory, w/o minimal intervention from the processor.

→ To initiate the transfer of a block of words, the processor sends the following data:

(i) starting address

(ii) word count

(iii) r/w mode

(iv) a control to start

Steps

→ Bus request (BR) is used by the DMAC (DMA controller) to request CPU to relinquish control of the buses.

→ CPU activates the bus grant (BG)

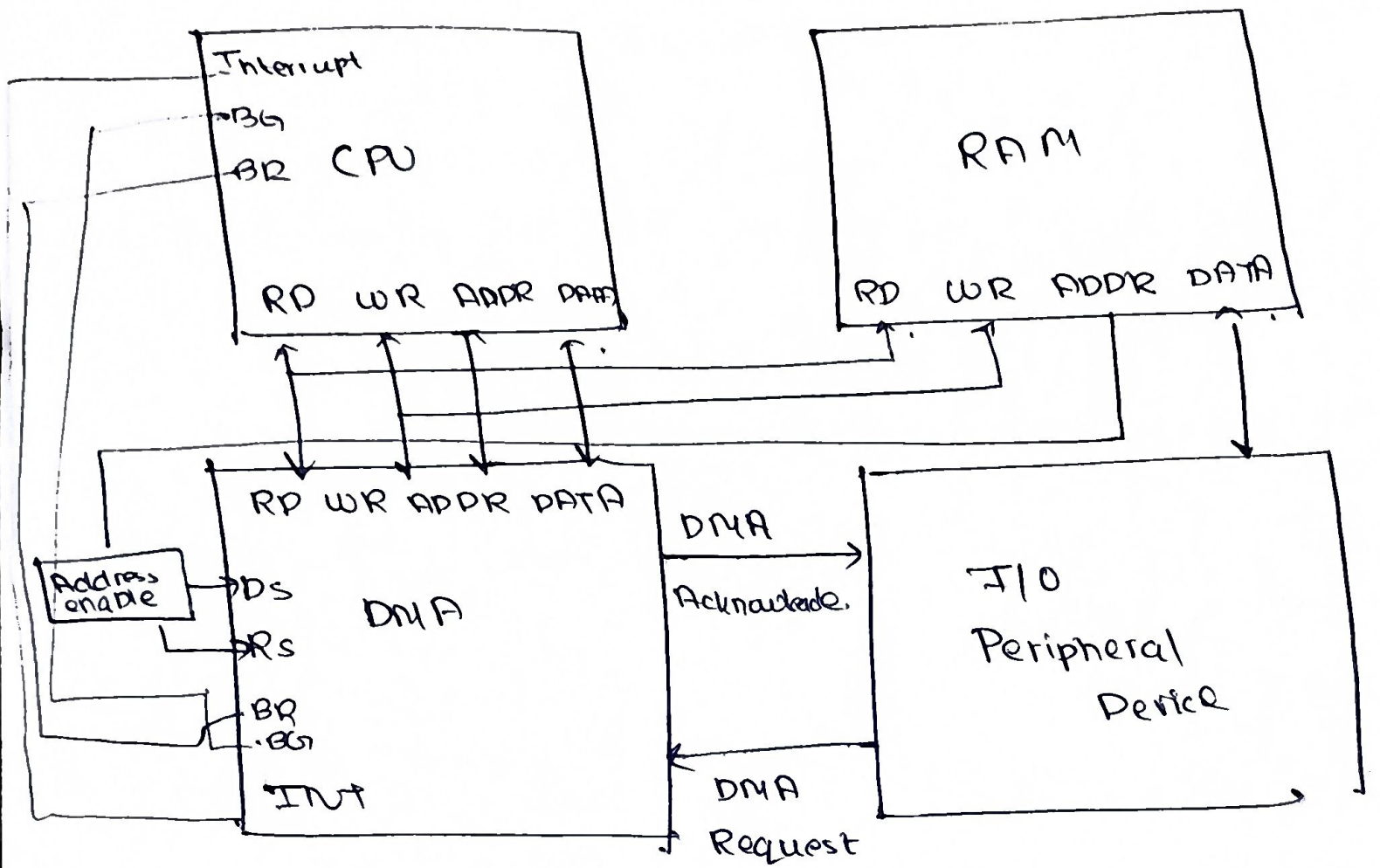
→ DMA takes control of the buses to conduct memory transfer w/o processor intervention.

DMA Transfer can be:

(i) Burst mode - a block sequence consisting of ano. of memory words transferred continuously

(ii) Cycle Stealing : allows DMA controller to transfer one data word at a time, after which control of buses is given back to the CPU

CPU delays ops for 1 clock cycle to allow the DMA transfer.



Request sent by peripheral

activate BR - relinquish control

CPU responds w/ BG line - buses are disabled.

DMA puts value of addr register in address bus

BG = 0 ⇒ RD, WR = Input

BG = 1 ⇒ RD, WR = Output

for every word, increments addr, decrements counter

word count = 0 ⇒ remove bus request

Informs CPU via an interrupt

* Bus Arbitration

Bus Master: Device that initiates data transfer on the bus

Bus Arbitration: The process by which the next device to become master is selected

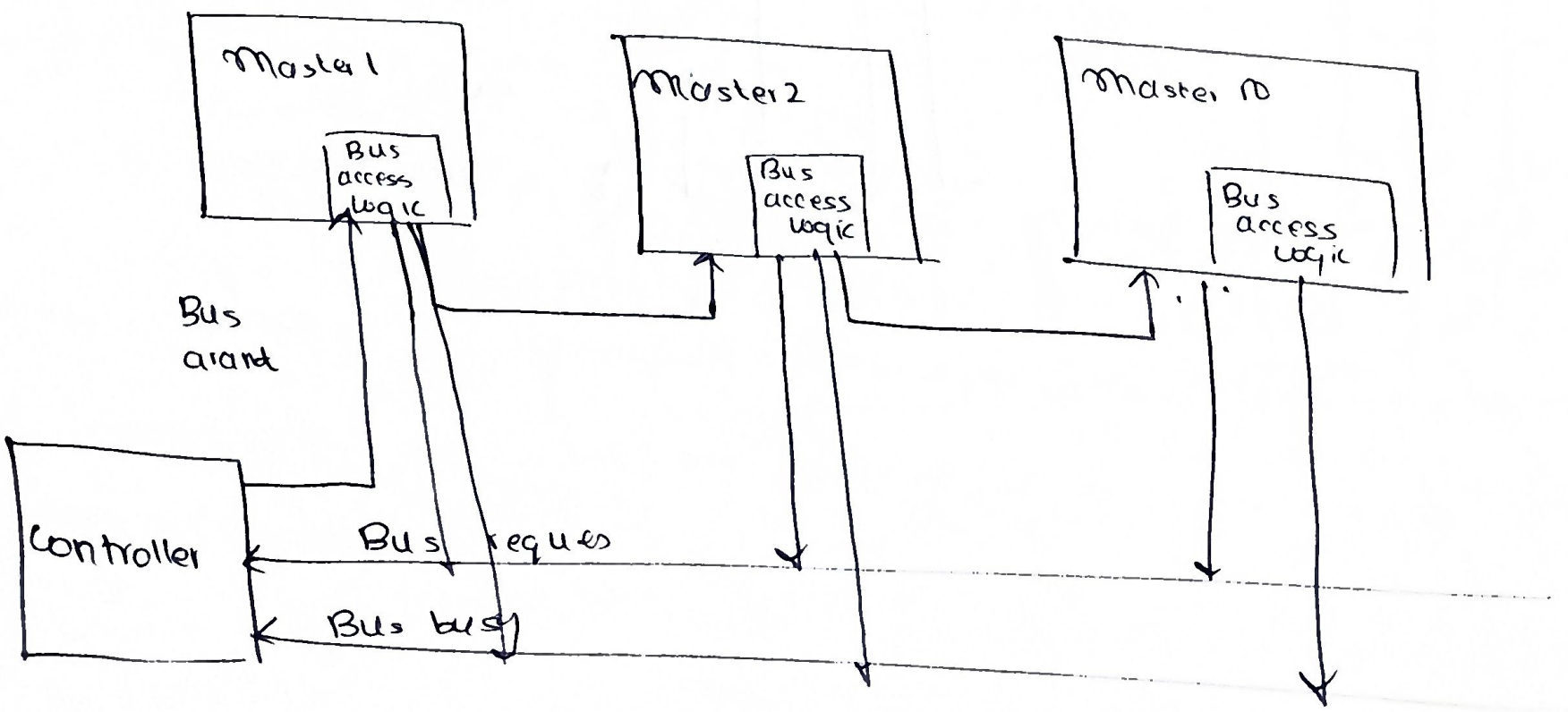
Types of Bus Arbitration: (i) Centralized

(ii) Decentralized (distributed)

Centralized Bus Arbitration

→ In this method, a single bus arbiter performs the required arbitration

→ Daisy Chaining - until the bus grant signal comes across the first master who is making a request for access, it travels serially through each master.



Distributed Arbitration

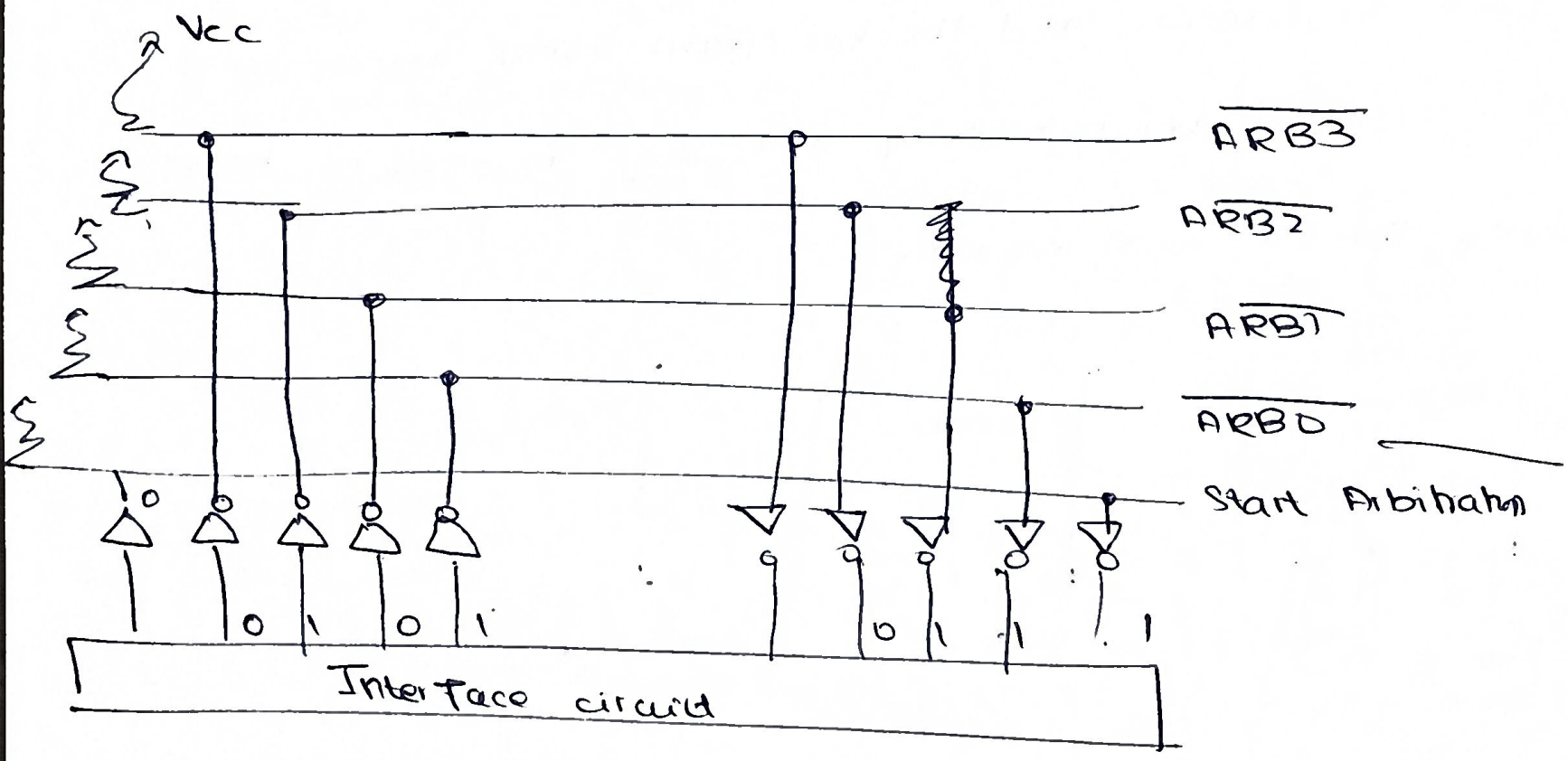
→ no single processor or controller controls the network.

→ all arbiters participate actively to decide who will be the bus master

→ Each device on the bus is given a 4 bit identification no.

→ One or more devices request the bus, and place their id no. on the collector lines

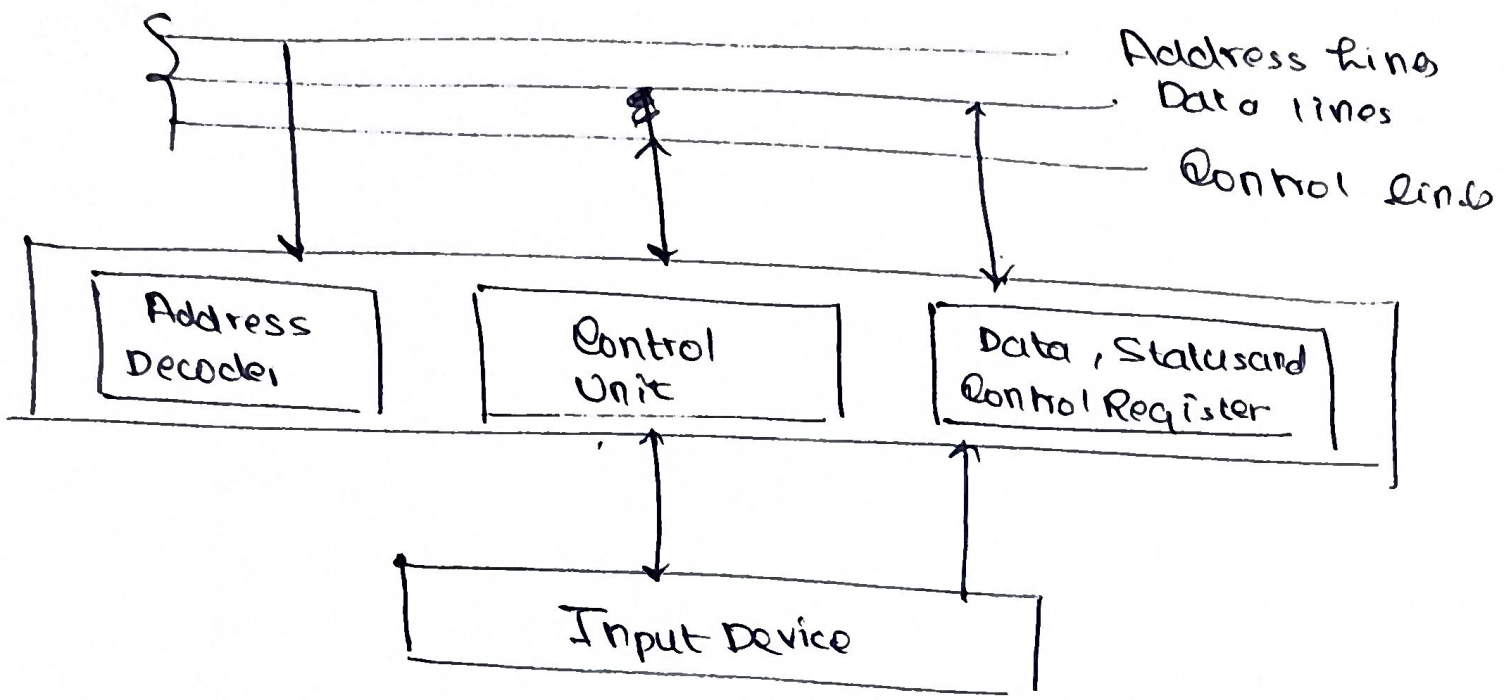
→ The device w/ the highest ID no. is chosen



* Interface Circuit

→ circuitry that is designed to link I/O devices to the processor.

→ acts as a mediator to make the computer communicate with the I/O modules



Features

---

has a data register: stores data temporarily while the data is exchanged between the I/O and processor

has a status register: bits indicate whether I/O device is set for transmission or not

has a control register: indicate type of op to be performed

User programs are prevented from using I/O operations directly because the OS does not provide direct access to the address space.