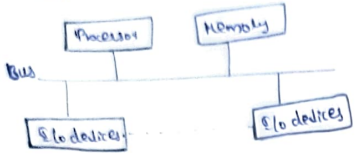
19/11/2023 Saturday

## * Accessing I/O devices

* Single-bus structure [A single bus connects all processor, memory & I/O devices]




A bus is formed through set of lines (data line, control line, address line).

A bus is a communication system that transfers data between components inside a computer, or between computers. [data bus, address bus, control bus]

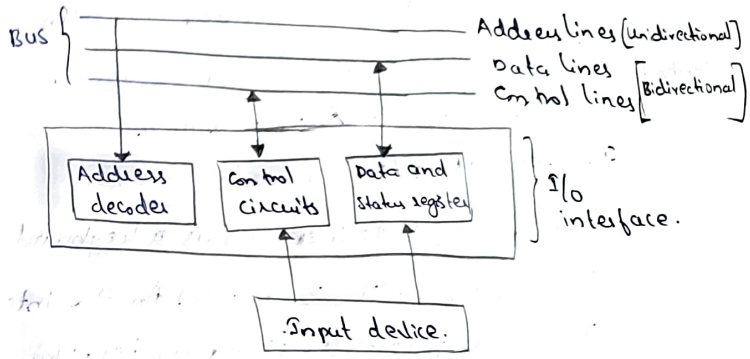
The bus enables all the devices connected to it to exchange information. Each I/O devices has assigned to a unique set of addresses.

When the processor places a particular address on the address lines, the device that recognizes this address responds to the commands issued on the control lines. The processor requests either a read or write operation, and the requested data are transfered over the data lines.

When I/O devices and the memory share the same address space, the arrangement is called memory-mapped I/O.

with memory mapped I/O, any machine instruction that can access memory can be used to transfer data to or from an I/O device.

## * I/O interface for an input device.




- **Address decoder**

  Enables the device to recognize it's address when this address appears on the address line.

- **Data Register**

  Holds the data being transfered to or from the Processor.

- **Status register**

  Contains information relevant to the operation of the I/O devices.

* Note: Both the data and status register are connected to the data bus and assigned unique addresses.

I/o devices operate at speeds that are vastly different from that of the processor.
[I/o devices are comparatively slow]

An instruction that reads a character from the keyboard should be executed only when a character is available in the input buffer of the keyboard interface. Also, we must make sure that an input character is read only once.

For an input devices such as a keyboard a status flag, SIN, is included in the interface circuit as part of the status register. This flag is set to '1' when a character is entered at the keyboard and cleared to '0' once this character is read by the processor.

Hence by checking the SIN flag, the software can ensure that it is always reading valid data. This is often accomplished in a program loop that repeatedly reads the status register and checks the state of SIN.

When SIN becomes equal to 1, the program reads the input data register. A similar procedure can be used to control output operation using an output status flag SOUT.

Program Controlled I/o, in which the processor repeatedly checks a status flag to achieve the required synchronization between the processor and an input or output device.

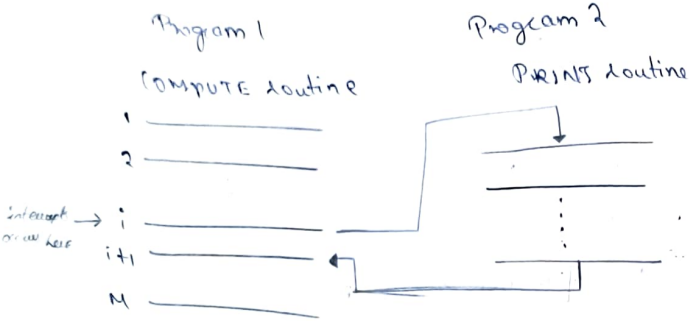
There are 2 other commonly used mechanisms for implementing I/o operations.

i) Interrupts
ii) Direct memory access (DMA)

## * Interrupts

Synchronization is achieved by having the I/o device send a special signal called interrupts, over the bus whenever it is ready for a data transfer operation. Atleast one of the bus control lines, called an interrupt-request line, is usually dedicated for this purpose. Since the processor is no longer required to continuously check the status of external devices, it can use the waiting period to perform other useful functions. The routine executed in response to an interrupt request is called the interrupt-service routine.

## Transfer of Control through the use of interrupts




Assume that an interrupt request arrives during execution of instruction 'i' in above figure. The Processor first completes execution of instruction i. Then it loads the program counter with the address of the first instruction of the interrupt-service routine, the processor has to come back to instruction i+1. Therefore, when an interrupt occurs, the current contents of the PC, which point to instruction i+1, must be put in temporary storage in a known location. A return from interrupt instruction at the end of the interrupt-service routine reloads the PC from that temporary storage location, causing execution to resume at instruction i+1.

The interrupt service routine may not have anything in common with the program being executed at the time the interrupt request is received. In fact, the two programs often belong to different users. Therefore, before starting execution of the interrupt-service routine, any information that may be altered during the execution of that routine must be saved. This information must be restored before execution of the interrupted program is resumed.

The information that needs to be saved and restored typically includes the condition code flags and the contents of any registers used by both the interrupted program and the interrupt-service routine.

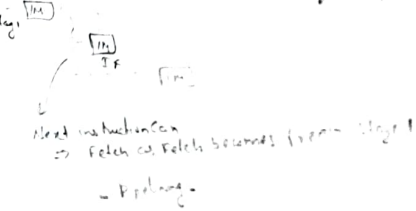
23/4/2023 (Thursday)

## Pipelining

- Faster execution of instruction
- whenever it's free, other process can execute. (processor)

*MIPS (5 stages)
1. Instruction Fetch (IF)
2. Read registers, instruction decode (ID) (Instruction decode)
3. execute (alu operation → performed) (Execute) EX
4. Result stored in memory. (Memory Access) ME
5. write result in memory (write Back) WB

Fig 4.34



- Structural hazard [waiting as hardware is not available]
- Data hazard [data inavailability]
  ↳ bypassing technique used
  After execution data given for next execution

- Control hazard
  if a jump instruction
  waited, after installing stall (idle).


24/11/2023 Friday

## DMA [Direct Memory Access]

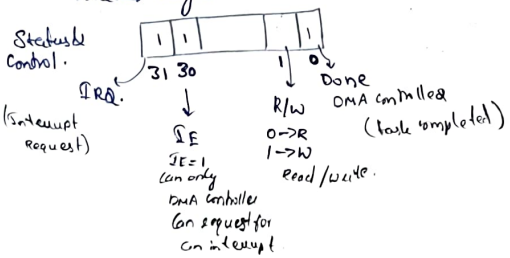
5th module
text-classroom.

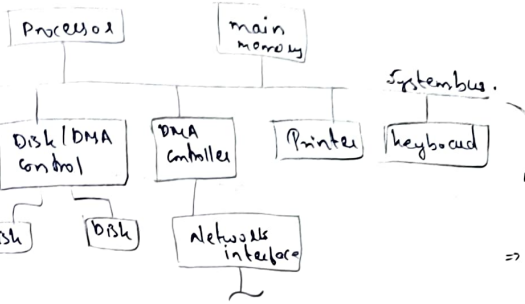
Processor Reading data from input device
↳ DMA Controller
Processor can do other tasks while DMA Controller performing reading (large memory access)
As processor do not have to spent all time for large data reading.

Processor → gives → start address, word count → to DMA Controller
Once DMA finishes tasks
Stops the (current working process (goes to blocked state)) gives an interrupt (telling the task has been finished.

- Blocked data - waiting for some data. (state)
  previously blocked data goes to running state. → state.
  (To reduce idle time of processor)

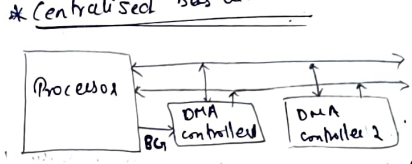
- Status register (32-bit) [for DMA controller]




## * Bus Arbitration

## * Centralised Bus arbitration.

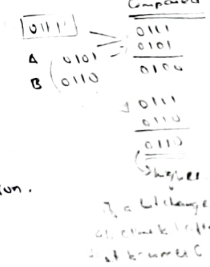



=> High priority DMA controller get the processor first.
Here DMA controller 1

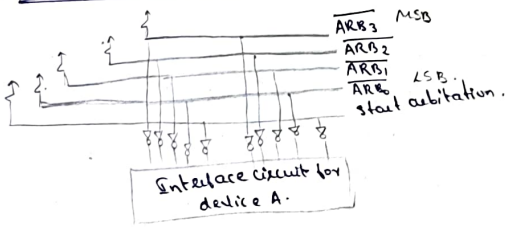
=> BG - Bus Grant Signal
indicates the BG granted the processor for DMA controller 1

To check whether bus is busy or not → Bus Busy line.

- Rotating priority → Once a bus is executed the priority changes.




## * Distributed Bus Arbitration



Module-5 [Pipelining...] Text 2

- Cache - In between processor and memory
  (registers) - fastest memory.
  Inside processor
  Frequently / recently used data are stored in cache

Smaller in size

Processor → Cache memory → Memory

= Locality of reference.

1. Temporal (time related) (loop data)
2. Spatial (space related)

- If required data present in cache - Cache hit - read hit, - write hit.
- Else Cache miss (data not in cache memory).
  - read miss
  - write miss

## * Cache mapping techniques

Decides where to place the data in the memory (location).
Which data to be replaced if cache is full - Cache Replacement algorithm is used.

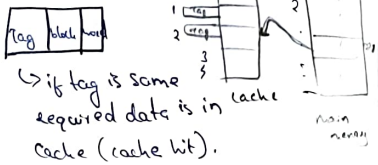
- Main memory will be divided into blocks (Larger memory)
- Cache will also be divided into blocks.

(Small memory) (divided as same no. of blocks with same size)
- Same block size -

2^7 / 128 = 2

## * Mapping techniques

- Direct Mapping
- Associative
- Set Associative




↳ if tag is same required data is in cache (cache hit).

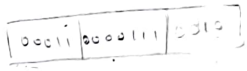
2^7 = 128

Q. 128 blocks, each block has 16 words, 16 bit addressable.

ans). 16 words => 2^4 => 4 bits
128 blocks => 2^7 => 7 bits
remaining 4+7=11, 16-11=5 bit.


16 bit.
remaining.

2^4 = 16
0-15.
16 words.
4 bits for addr used.


=> 2^16 memory
64k memory size
2^16 = 64k.
16 bit address.

## * Direct Mapping

Q. Consider a direct mapped cache of size 16 kB, with block size 256 bytes the size of main memory is 128 kB.

i) Find the no. of bits in tag.
ii) Tag directory size.


block size = 256 bytes = 2^8 => 8 bits for word
Main memory size = 128 kB
Cache memory size = 16 kB

2^10 x 2^7 = 2^17 = 128 kB.

No. of blocks = Cache size / block size = 16kB / 256 = 2^14 / 2^8 = 2^6 blocks

Tag = 17 - [6 + 8]
= 3 bits.

Tag directory size = No. of blocks x 3
= 2^6 x 3 = 64 x 3 = 192 bits = 24 bytes.

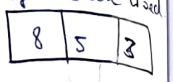
Q. Consider a machine with a byte addressable main memory of $2^{16}$ bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines used with this machine. How many bits will be there in tag line/block and word field, [block/line] of format of main memory addresses.


ans): Main memory size = $2^{16}$ bytes

(word) block size = 8 bytes $\Rightarrow 2^3$

No. of blocks/lines = 32

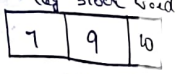
$\Rightarrow 2^5$




8 bits for tag
5 bits for block
3 bit for word.

16 - 8 = 8

Q. Consider a direct mapped cache of size 512 kB with block size 1 kB. There are 7 bits in the tag. Find i) The size of main memory
ii) Find the size of tag directory.




ans): Cache memory size = 512 kB $= 2^{19}$

Block size = 1 kB $= 2^{10}$

No. of bits in tag = 7

No. of blocks/lines = $\frac{\text{Cache size}}{\text{block size}} = \frac{2^{19}}{2^{10}} = 2^9$

1 kB = $2^{10}$
1 MB = $2^{20}$
1 GB = $2^{30}$

2 powers.

i) Main memory size = 7 + 9 + 10 = $2^{26}$ = $2^6 \times 2^{20}$ = 64 MB

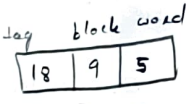
ii) Tag directory = No. of block lines × No. of bits in tags

$= 2^9 \times 7 = 512 \times 7$ bits

$= \frac{512 \times 7}{8}$ bytes

$= 2^6 \times 7 = 64 \times 7 = 448$ Bytes

Q. Consider a machine with a byte addressable main memory of $2^{32}$ bytes divided into blocks of size 32 bytes. Assume that a direct mapped cache having 512 cache lines is used with this machine, the size of the tag field in bits is _____.




ans): No. of cache lines = $512 = 2^9$

Main memory size = $2^{32}$ bytes

Block size = 32 bytes $2^5 = 32$

$\Rightarrow$ word

32 - 14 = 18

Q. An 8 kB direct map write back cache is organized as multiple blocks each of size 32 bytes the processor generates 32 bit addresses, the cache controller maintains the tag information for each cache block comprising of the following:

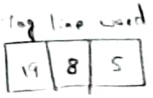
1) write back
2) write through
Dirty bit - 1 value changed

1 valid bit and 1 modified bit as many bits as the minimum needed to identify the memory block mapped in the cache. What is the total size of memory needed at the cache controller to store meta data (tags) for the cache

$\Rightarrow$ dirty bit is needed as it is a write back cache

ans): Cache size = 8 kB = $2^{13}$

$2^{10} \times 2^3 \Rightarrow 2^{13}$




Block size = 32 bytes = $2^5$ word

Total length = 32 bit.

No. of lines = $\frac{\text{Cache size}}{\text{block size}} = \frac{2^{13}}{2^5} = 2^8$

No of lines × bits in tag in block

Tag directory = $2^9 \times 19 + 2$

$= 256 \times 21$

$= 5376$

256, 21, 256, 512, 5376


## Set Associative mapping


- one block is divided into n way sets.

Q. Consider a two way set associative mapped cache of size 16kB with block size 256 bytes the size of main memory 128 kB. Find i) No. of bits in tags ii) and tag directory size

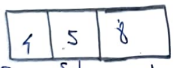
i) blocksize = 256 bytes $\Rightarrow 2^8$

main memory size = 128 $= 2^{17}$

Cache size = 16kB = $2^4 \times 2^{10} = 2^{14}$ (kB conversion)

$\Rightarrow$ n blocks – 1 set

$\Rightarrow$ 2 blocks – 1 set (2 way set associative mapping)




No of blocks = $\frac{2^{14}}{2^8} = \frac{2^6}{2} = 2^5$ (2 way)

ii) Tag directory = No. of blocks × Tag bit.

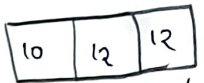
= $2^6 \times 4 = 2^8$ (No. of blocks..)

$2^6 \times 2^2 = 2^8$

Q. Consider a 4-way set associative mapped cache with blocksize 4kB, the size of main memory is 16GB and there are 10 bits in the tag find the size of cache memory and tag directory size.

Blocksize = 4kB = $2^{10} \times 2^2 = 2^{12}$

Tag bits = 10




1GB = $2^{30}$, 1MB = $2^{20}$, Size of main memory = 16GB = $2^{30} \times 2^4 = 2^{34}$.

No. of blocks = $\frac{\text{Cache size}}{\text{blocksize}}$ = $\frac{2^{34}}{2^{12}} = 2^{22}$

No. of blocks = $2^{12} \times 4 = 2^{12} \times 2^2 = 2^{14}$ (4 way set associative)

Cache size = No. of blocks × blocksize.

= $2^{14} \times 2^{12} = 2^{26} \Rightarrow 2^{20} \times 2^6$

= 64MB

tag directory = No. of blocks × Tag bit.

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

= ~~$\frac{2^{20} \times 2^4}{2^{20} = 1MB}$ = 16MB~~

$2^8 \times 5$

$2^{15} \times 5$

$\frac{2^{15}}{2^3} \times 5$ (÷ by 4 bytes)

$2^{12} \times 5$

$4 \times 5 \times 2^{10}$

20 kB

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

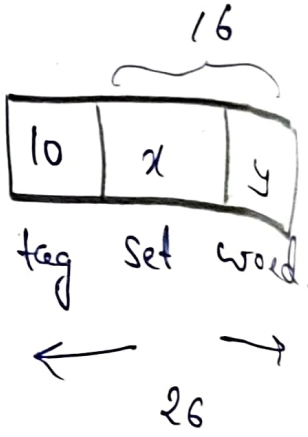
$16 \times 10 \times 2^{10}$

160 KB

$160/8 = 20$ KB

$2^{10} \Rightarrow 1$ KB

(2.) Consider a 8-way set associative mapped cache the size of cache memory is 512 kB, and there are 10 bits in the tag. Find the size of main memory.




Cache memory size = 512 kB

$\Rightarrow 2^9 \times 2^{10} \Rightarrow 2^{19}$

No. of bits in tag = 10.

Cache size = No. of Sets x 8 x block size.

$= 2^x + 2^3 + 2^y$

$2^{19} = 2^{3+x+y}$

$3 + x + y = 19$

$x + y = 16$

Main memory size = $2^{26} \Rightarrow 2^{20} \times 2^6$

$= 2^6$ kB

$=$ 64 KB