

Computing Systems Organization

CT101 – Computing Systems

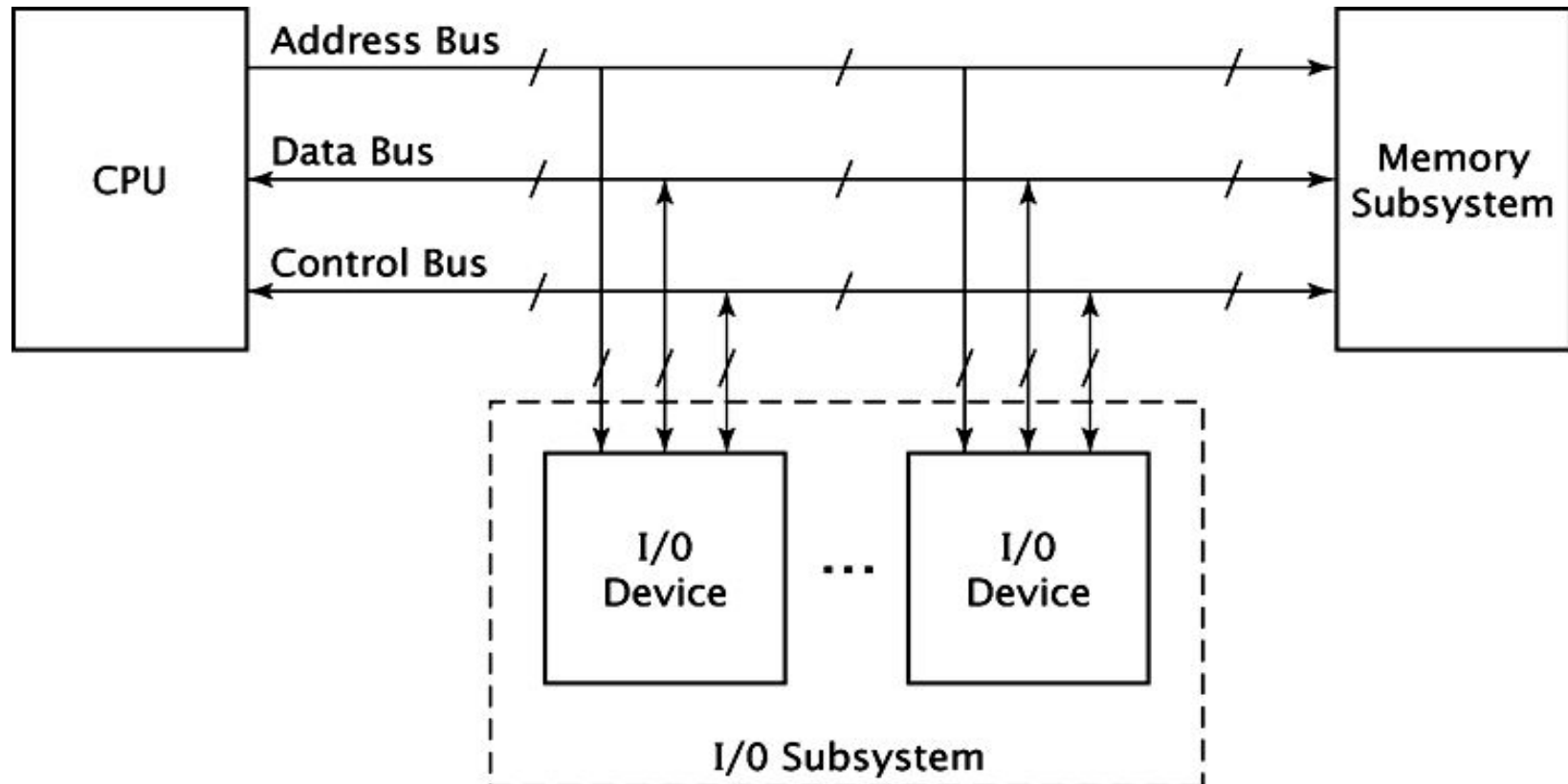
Contents

- Describe the operation of a computer at the functional level
- Explain the function of the main components of a computer system
- Detail the interaction of computer sub-systems
- Detail the organization of computer sub-systems

Computer organization

- Instruction set architecture provides a good understanding of what a microprocessor can do; provides no information how to use the processor in a bigger system.
- In order to design a computing system, more information is needed than the instruction set
- Computer system subsystems:
 - CPU
 - Buses
 - Memory
 - Input/Output

Basic Computing Systems Organization



System buses

- Set of wires, that interconnects all the components (subsystems) of a computer
 - A source component sources out data onto the bus
 - A destination component inputs data from the bus
- May have a hierarchy of buses
 - Address, data and control buses to access memory and an I/O controller.
 - Second set of buses from I/O controller to attached devices/peripherals
 - PCI bus is an example of a very common local bus

Address bus

- CPU reads/writes data from the memory by addressing a unique location; outputs the location of the data (aka address) on the address buss; memory uses this address to access the proper data
- Each I/O device (such as monitor, keypad, etc) has a unique address as well (or a range of addresses); when accessing a I/O device, CPU places its address on the address bus. Each device will detect if it is its own address and act accordingly
- Devices always receive data from the CPU; CPU never reads the address buss (it is never addressed)

Data bus

- When the CPU fetches data from memory, it first outputs the address on the address bus, then the memory outputs the data onto the data bus; the CPU reads the data from data bus
- When writing data onto the memory, the CPU outputs first the address on the address bus, then outputs the data onto the output bus; memory then reads and stores the data at the proper location
- The process to read/write to a I/O device is similar

Control bus

- Address and data buses consist of n lines, which combine to transmit one n bit value; control bus is a collection of individual control signals
- These signals indicate whether the data is to be read into or written out the CPU, whether the CPU is accessing memory or an IO device, and whether the I/O device or memory is ready for the data transfer
- This bus is mostly a collection of unidirectional signals

CPU

- The Central Processing Unit (CPU a.k.a. Processor) is the chip which acts as a control center for all operations. It executes instructions (a program) which are contained in the memory section.
- Basic operations involve
 - the transfer of data between itself and the memory section
 - manipulation of data in the memory section or stored internally
 - the transfer of data between itself and input/output devices
- The CPU is said to be the **brains** of any computer system. It provides all the timing and control signals necessary to transfer data from one point to another in the system.

Programs: Instructions and Operands

- A program = a number of CPU instructions.
- Instruction components:
 - an instruction code (aka OPCODE)
 - one or more operand's (data which the instruction manipulates)

Address		
000	MULT A, #02	Instruction
001	02	Operand
002	DEC A	Instruction
003	ADD A, #fe0f	Instruction
004	0f	Operand
005	fe	Operand
006		Instruction
007		

The instruction specifies to the CPU what to do, where the data is located, and where the output data (if any) will be put.

Instructions are held in the memory section of the computer system. Instructions are transferred one at a time into the CPU, where they are decoded then executed. Instructions follow each other in successive memory locations.

Memory locations are numbered sequentially. The processor unit keeps track of the instruction it is executing by using a internal counter (location in the memory) known as the *program counter* (sometimes called *instruction pointer*).

Von Neumann and Harvard architectures

- Von Neumann
 - Allows instructions and data to be mixed and stored in the same memory module
 - More flexible and easier to implement
 - Suitable for most of the general purpose processors
- Harvard:
 - Uses separate memory modules for instructions and for data
 - It is easier to pipeline and there are no memory alignment problems
 - Higher memory throughput
 - Suitable for DSP (Digital Signal Processors)

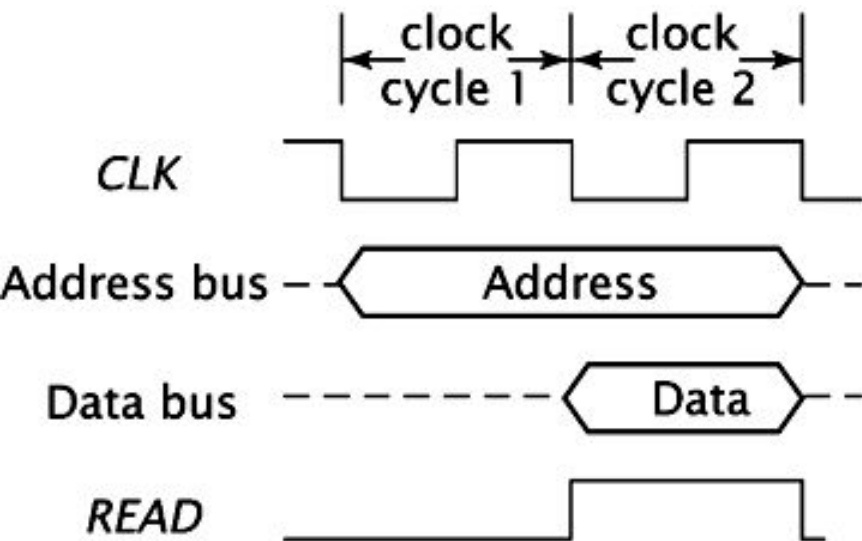
Computer Memory

- Memory contains instructions for the processor to execute or data it operates on
- **Address Locations** - Memory consists of a sequential number of locations, each of which are a specific number of bits wide.
 - byte wide memory 8 bits (PC-8088)
 - 16 bits (XT-8086, AT-80286)
 - 32 bits (386DX, 486SX, 486DX)
 - 64 bits (Modern systems – Pentium and up)
- Each memory location is referred to as an **address**, and generally expressed in hexadecimal notation (using base 16 numbers).
- The size is denoted as the number of locations times the number of bits in each location
- The processor selects a specific address in memory by placing the address on the **address bus** . The value on this address bus is used by the memory system to find the data at the specific location

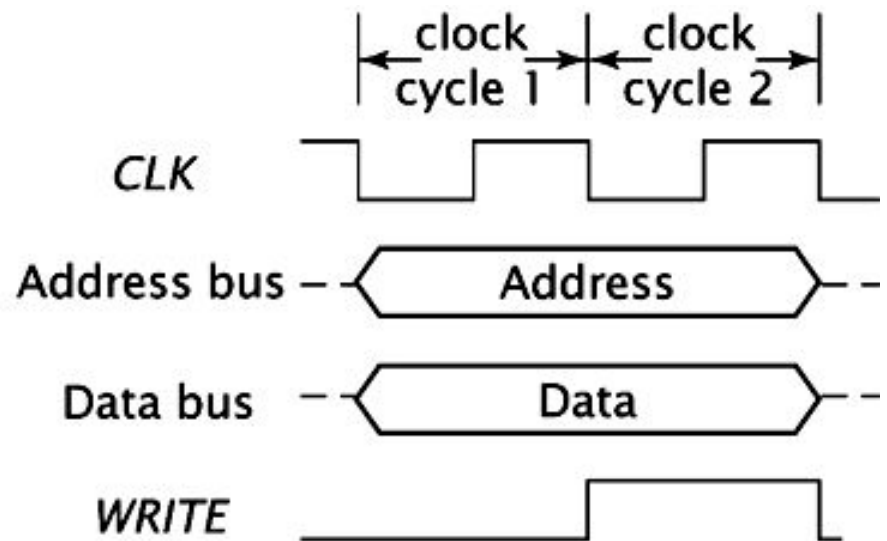
Computer Memory

- The total number of address locations which can be accessed by the processor is known as its **physical address space**. How large this is determined by the size of the address bus, and is often expressed in terms of Kilobytes (x1024), Megabytes or Gigabytes.
 - 16 bits address bus = 64K (65536 locations)
 - 20 bits address bus = 1MB (IBM PC)
 - 32 bits address bus = 4GB (486DX)
- **Access Times** - Access time refers to how long it takes the processor to read or write to a specific memory location within a chip. The limiting factor is the type of technology used to implement the memory cells inside the chip.
- **Volatility** - This refers to whether or not the contents of the memory is lost when power is turned off. If the contents are lost, the memory is *volatile*. If the contents are retained, then the memory is *non-volatile*.

Memory read/write operations



(a)



(b)

- a) Memory read operation
- b) Memory write operation

Input/Output devices

- Peripheral devices allow input and output to occur.
- Examples of peripheral devices are
 - disk drive controllers
 - keyboards
 - mice
 - video cards
 - parallel and serial cards
 - real-time clocks
- The processor is involved in the initialization and servicing of these peripheral devices.

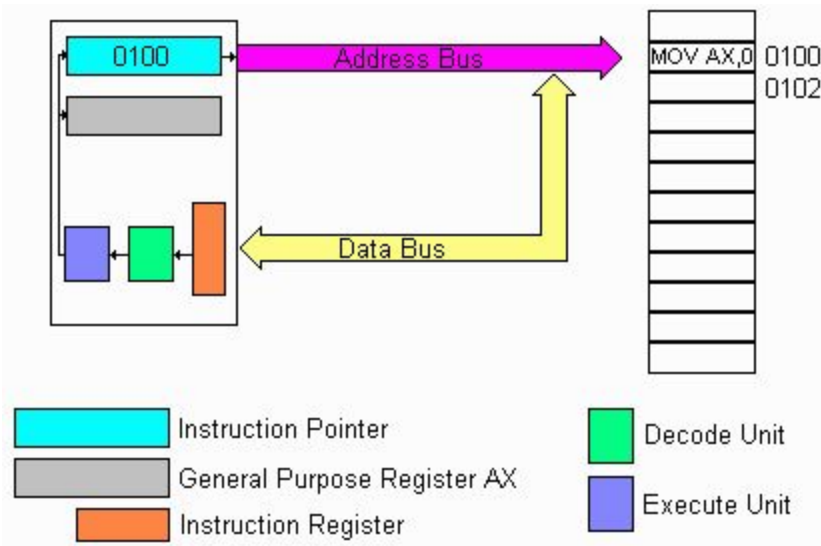
I/O read/write operations

- The I/O read and write operations are similar to the memory read and write operations.
- A processor may use:
 - **memory mapped I/O** (when the address of the I/O device is in the direct memory space, and the sequence to read/write data in the device are the same with the memory read/write sequence)
 - **isolated I/O** – the process is similar, but the processor has a second set of control signals to make the distinction between a memory access and an I/O access (memory locations and I/O devices can be located at the same address, which makes this extra control signal necessary); for I/O operations, the processor holds IO/M' (or similar) signal high for the duration of the I/O operation

CPU Function – Instruction cycle

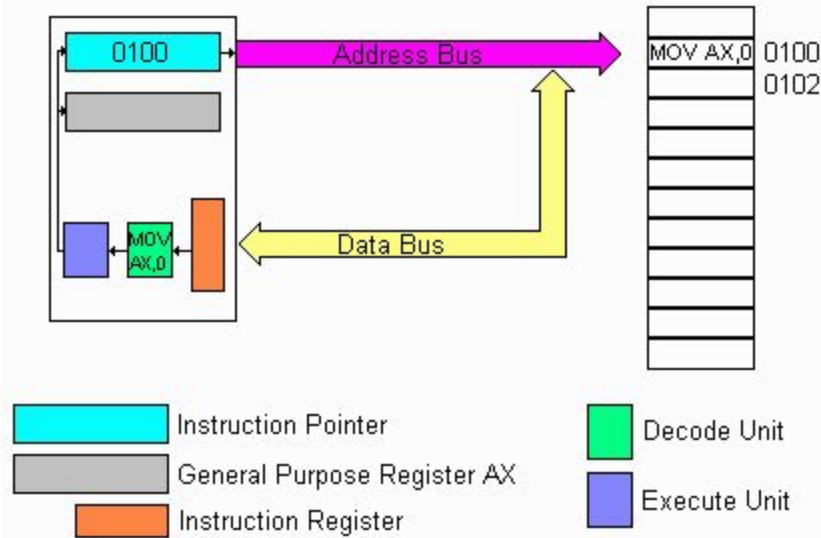
- The instruction cycle is the procedure of processing an instruction by the microprocessor:
 - Fetches (reads) the instruction from the memory
 - Decodes the instruction, determining which instruction is to be executed (what instruction has been fetched)
 - Executes the instruction – performs the operations necessary to complete what the instruction is suppose to do (different from instruction to instruction, may read data from memory, may write data to memory or I/O device, perform only operations within CPU or combination of those)
- Each of the functions fetch -> decode -> execute consist of a sequence of one or more operations inside the CPU (and interaction with the subsystems)

Fetch Cycle



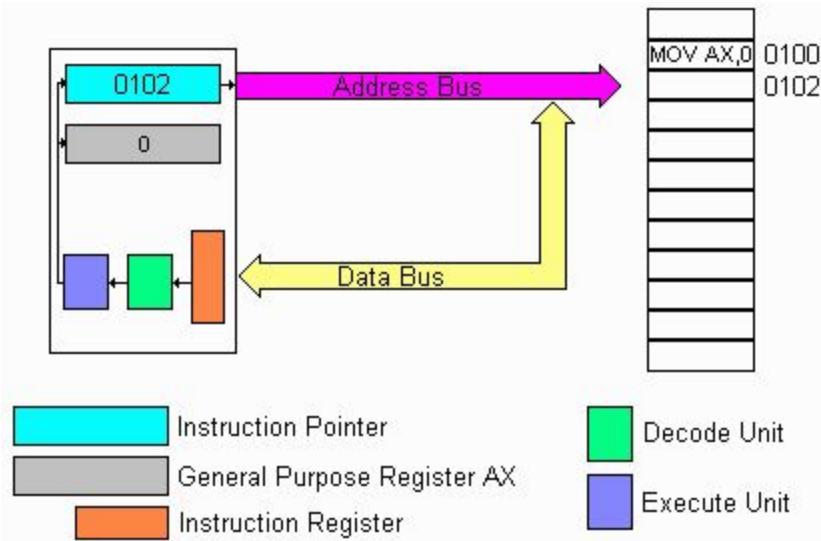
- In the first phase, the processor generates the necessary timing signals to fetch the next instruction from the memory system.
- The instruction is transferred from memory to an internal location inside the processor (the instruction register)
- In the above image, the processor is ready to begin the Fetch cycle. The current contents of the **instruction counter (program counter)** is address 0100. This value is placed on the address bus, and a READ signal is activated on the control bus. The memory receives this and finds the contents of the memory location 0100, which happens to be the instruction MOV AX, 0.
- The memory places the instruction on the Data Bus, and the processor then copies the instruction from the Data Bus to the Instruction Register.

Decode Cycle



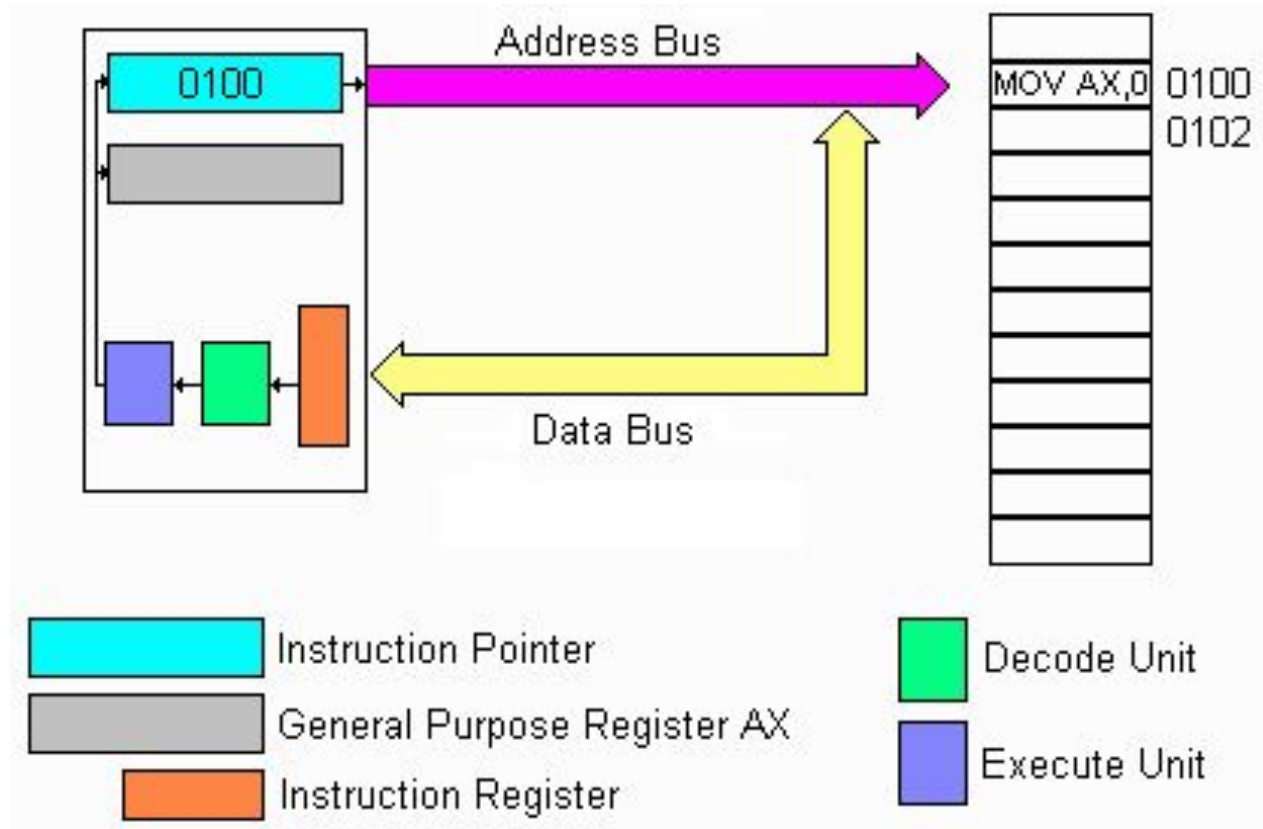
- The processor transfers the instruction from the instruction register to the Decode Unit.
- It compares the instruction to an internal table, and when a match is found, the table contains the list of macro instructions (a number of steps) which are required to perform the instruction.
 - In our case, the instruction means place the value 0 into the AX register. The decode unit now has all the details of how to do this.
- During this phase the processor (if required by the instruction) will get any operand's required by the instruction.
 - The final effect of instruction MOV AX, 0 is to set the value of the AX register of the processor to the constant value 0. The processor has the instruction (MOV AX), but now needs the constant value 0 to complete the instruction before executing it. In this instance, the processor will fetch the constant value 0 from the next location in memory (it is found immediately after the instruction, in the next memory location 0101)

Execute Cycle



- In the last phase, the processor executes the instruction. In the example above, this involves setting the contents of the internal register AX to the constant value 0
- The final part of execute phase is to adjust the **Instruction Counter** to point to the next instruction to be executed, which is found at address 0102

Fetch/Decode/Execute



Animated fetch decode execute

References

- “Computer Systems Organization & Architecture”, John D. Carpinelli, ISBN: 0-201-61253-4
- “Operating Systems – A modern perspective”, Garry Nutt, ISBN 0-8053-1295-1