Chapter 3 – A Top-Level View of Computer Function and Interconnection

Computer Components & Function (Section 3.1 & 3.2) on pp. 55-73

Let's begin by taking a look at a top-level view of the computer components:

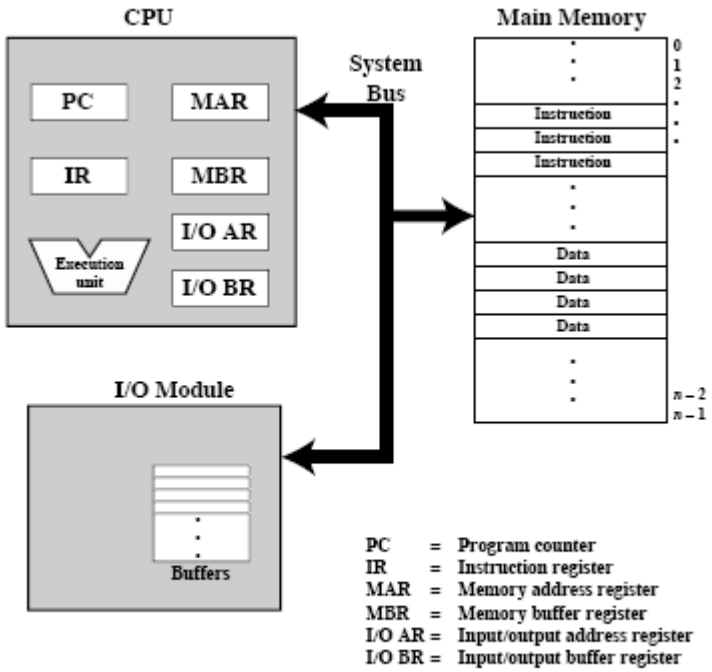


Figure 3.2 Computer Components: Top-Level View

The basic function of a computer is to execute a program by executing a set of instructions held in memory. To understand the execution of a program, we must first understand the basics of an instruction cycle as it relates to Figure 3.2 above.

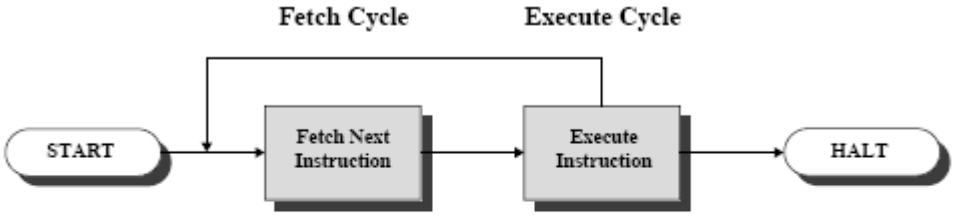


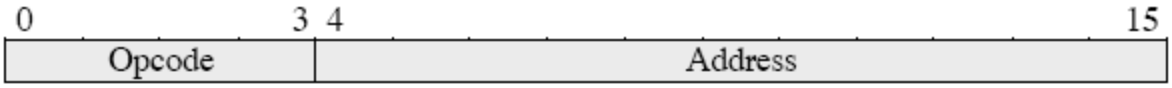
Figure 3.3 Basic Instruction Cycle

P#1: Explain an instruction fetch using the components of Figure 3.2.
1) The PC holds the address of the next instruction to execute. The contents of the PC are placed on the System Bus and the PC is incremented to the next instruction to be executed.
2) The instruction from Main Memory is retrieved and placed into the IR using the System Bus.

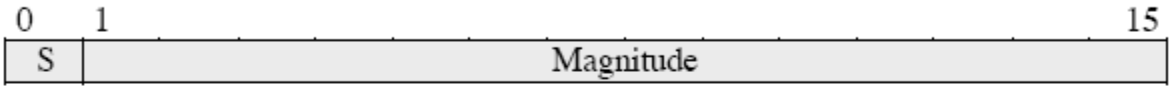
Note: The MAR and MBR registers are also used in the process but for now we will ignore their use for simplicities sake.

The processor will then interpret the instruction and perform an action. What are these possible actions?

Consider a hypothetical computer with a single accumulator (register) where both instructions and data are 16-bits long.



(a) Instruction format



(b) Integer format

Program Counter (PC) = Address of instruction
Instruction Register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from Memory
0010 = Store AC to Memory
0101 = Add to AC from Memory

(d) Partial list of opcodes

Figure 3.4 Characteristics of a Hypothetical Machine

Q#1: How many instructions does this hypothetical computer have? Why?

P#2: Give the hexadecimal representation for the number 15 and -15.

What are the steps that will be taken to execute the first three instructions below using the hypothetical computer described on the previous page?

Remember 0001 = Load AC from memory, 0010 = Store AC to memory, 0101 = Add to AC from memory

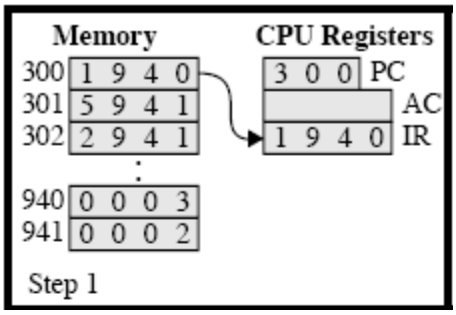


Figure 3.5 Example of Program Execution (contents of memory and registers in hexadecimal)

Interrupts

Interrupt - is an external request for service. In particular, an interrupt causes the microprocessor to stop executing the current procedure (saving the status) and continue on with the routine specified by the interrupt. When the interrupt has been fully serviced, control returns to the previously executing routine.

Two types of interrupts exist:
1) maskable - depending on the status of the interrupt flag, this interrupt can be ignored by the hardware.
2) nonmaskable - must be acknowledged by the hardware independent of the interrupt flag.

Q#2: What are examples of maskable and nonmaskable interrupts?

In general, an instruction cycle with interrupts looks like the following:

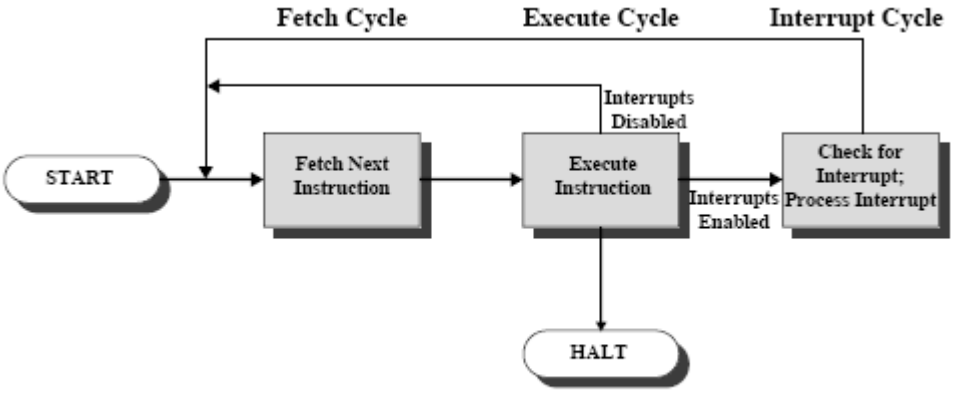


Figure 3.9 Instruction Cycle with Interrupts

Therefore, the flow of a program looks like the following with interrupts:

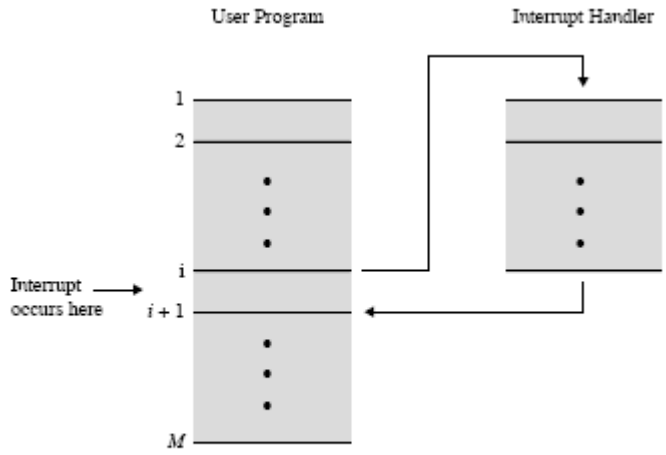


Figure 3.8 Transfer of Control via Interrupts

Note: In figure 3.8, the instruction completes at location i and then any pending interrupts are checked for by the processor. If an interrupt is found as in the case of figure 3.8, the next instruction at i+1 is saved and execution continues at the interrupt handler. Once the interrupt handler has completed, control returns to the instruction at location i+1.

The complete instruction cycle with interrupts is as follows:

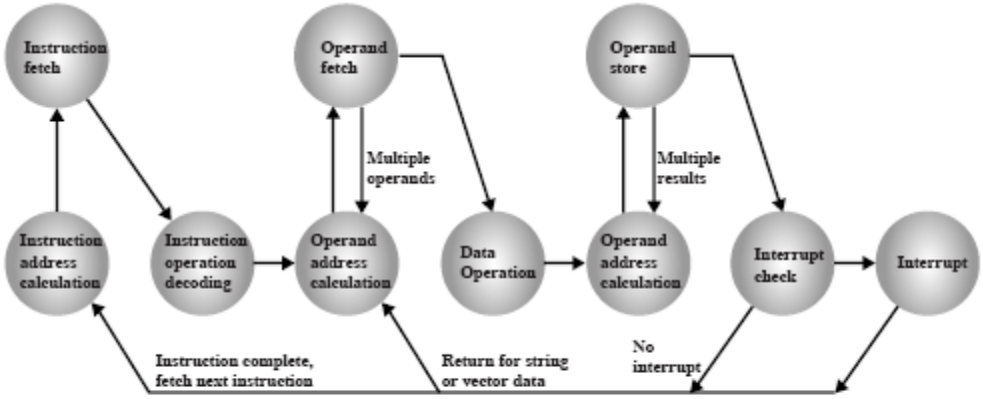


Figure 3.12 Instruction Cycle State Diagram, With Interrupts

Interconnection Structures & Bus Interconnection (Section 3.3 & 3.4) on pp. 73-84

We now move to the interconnection structures that exist between the various modules or components of a computer. If we look at the basic module types, we see the following:

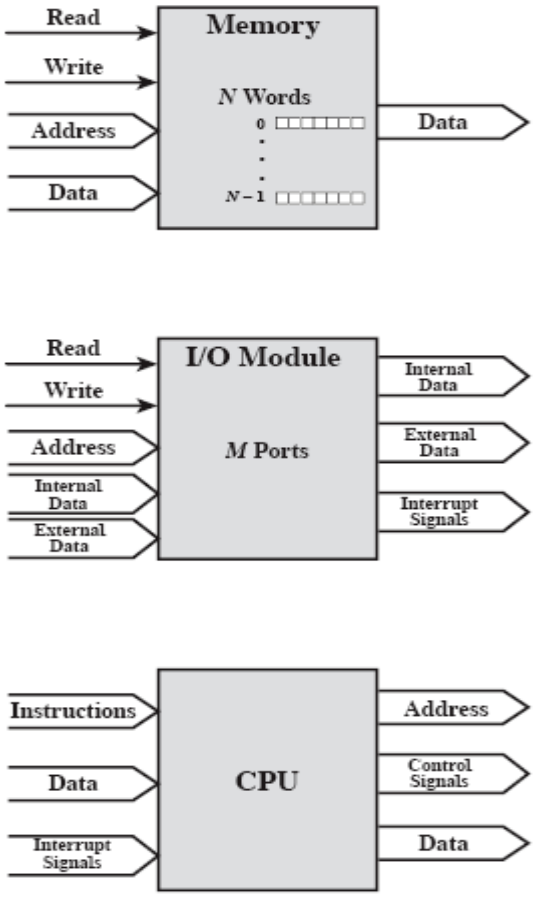


Figure 3.15 Computer Modules

Given these modules, the interconnection structure must support the following data exchanges:

- Memory to Processor
- Processor to Memory
- I/O to Processor
- Processor to I/O
- I/O to Memory
- Memory to I/O

Note: I/O to/from Memory can be accomplished with processor intervention using direct memory access (DMA).

Bus – communication pathway connecting two or more devices.

Multiple buses exist in a computer system. A bus that connects major computer components (Processor, Memory, and I/O) is called a system bus.

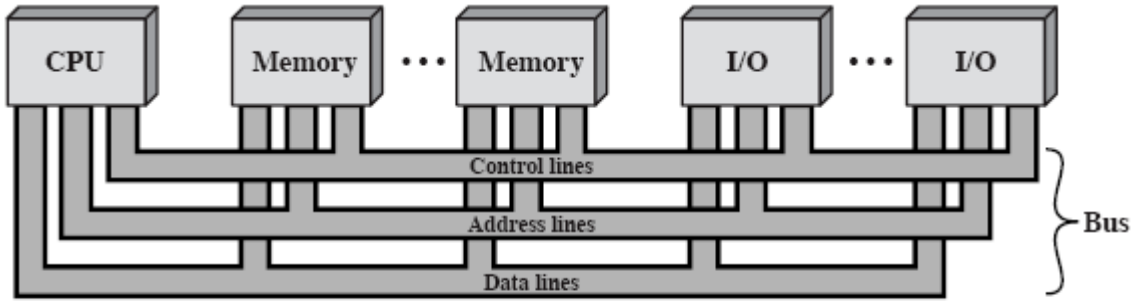


Figure 3.16 Bus Interconnection Scheme

Q#3: In today's modern computers, how many bits would you say make up the address bus? data bus?

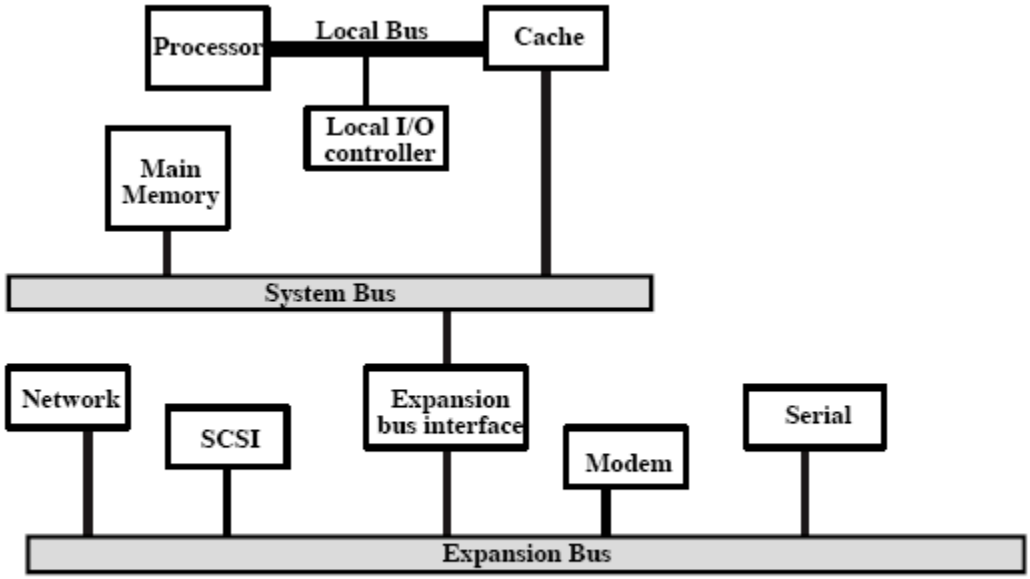
Q#4: Given the size you supplied in Q#3 for the address bus, how much memory can be accessed? Why?

Q#5: What is the control bus used for?

Control lines include:

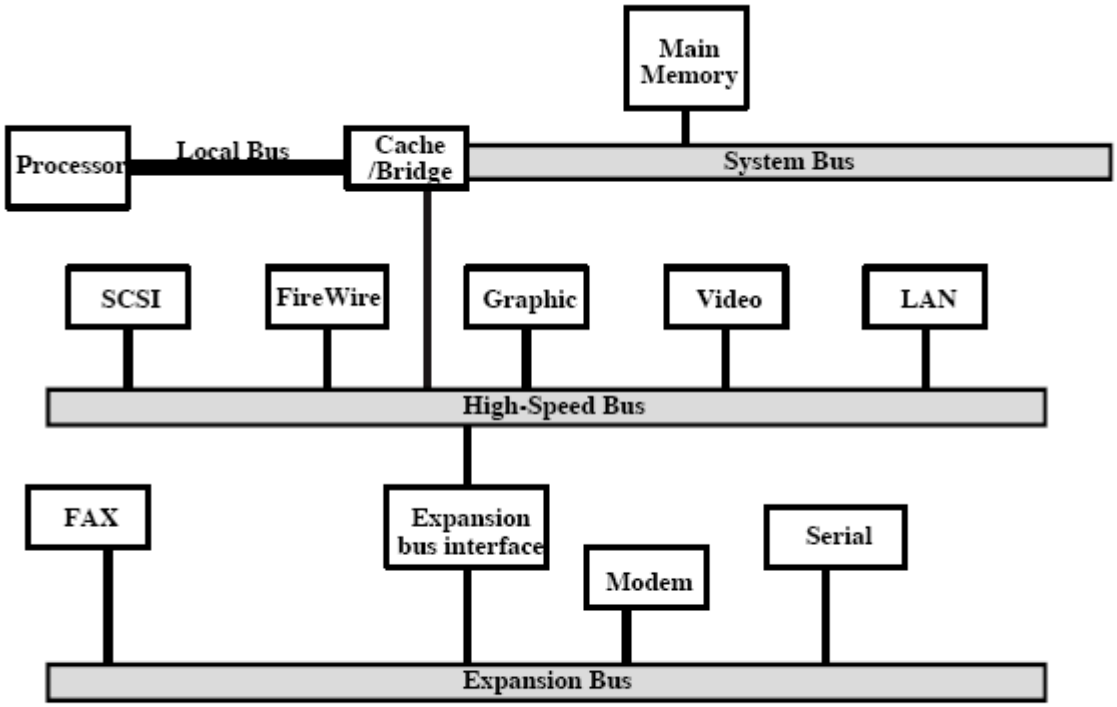
- Memory Write
- Memory Read
- I/O Write
- I/O Read
- Transfer Acknowledge
- Bus Request
- Bus Grant
- Interrupt Request
- Interrupt Acknowledge
- Clock
- Reset

Traditional bus architectures look like the following:



(a) Traditional Bus Architecture

As I/O devices have achieved a higher performance, this traditional architecture does not deliver an optimum performance because the speed of the expansion bus becomes a bottleneck. Better throughput can be obtained with the following architecture.



(b) High-Performance Architecture

This arrangement brings higher speed devices into closer integration with the processor without processor intervention. Slower devices are farther away.