Computer Architecture

Program Execution and Instruction Sets

INFO 2603
Platform Technologies

Week 2

Instruction Cycle

The basic function performed by a computer is the execution of a program, which is a set of instructions stored in memory.

The processor does the work of executing these instructions by (1) reading (fetch) the instructions from memory one at a time, and then (2) executing these instructions.

Program execution halts only when the machine is turned off, some sort of unrecoverable error occurs, or a program instruction that halts the computer is encountered.

Instruction Cycle

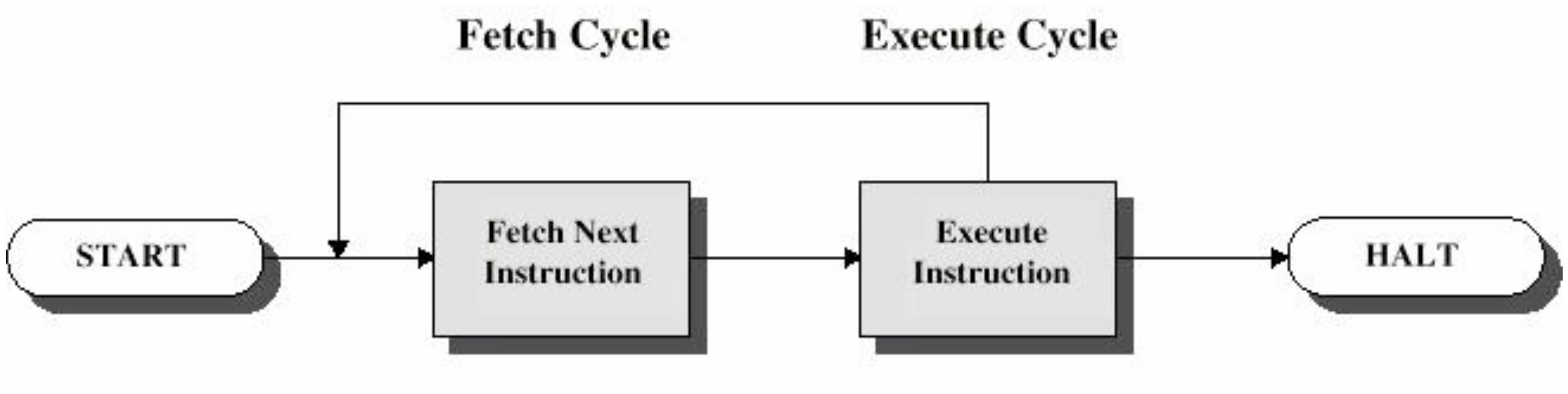


Figure 1. Basic Instruction Cycle

Interrupts

All computers provide a mechanism by which other modules (Input/Output, memory) may interrupt the normal processing of the processor.

Interrupts are meant to improve processing efficiency.

Most external devices are much slower than the processor which may have to pause and remain idle while waiting for such devices to “catch up”.

The length of this pause involves a lot of instruction cycles (hundreds or thousands) that are wasted on irrelevant operations.

Interrupts

Classes of interrupts include:

- Program: generated as a result of an instruction execution e.g. reference outside user’s allowed memory space, division by zero.
- Timer: generated by a timer within the processor. This allows the OS to perform certain functions on a regular basis.
- I/O: generated by an I/O controller to signal normal completion of an operation or to signal a variety of error conditions.
- Hardware Failure: generated by a failure such as power failure.

Instruction Cycle with Interrupts

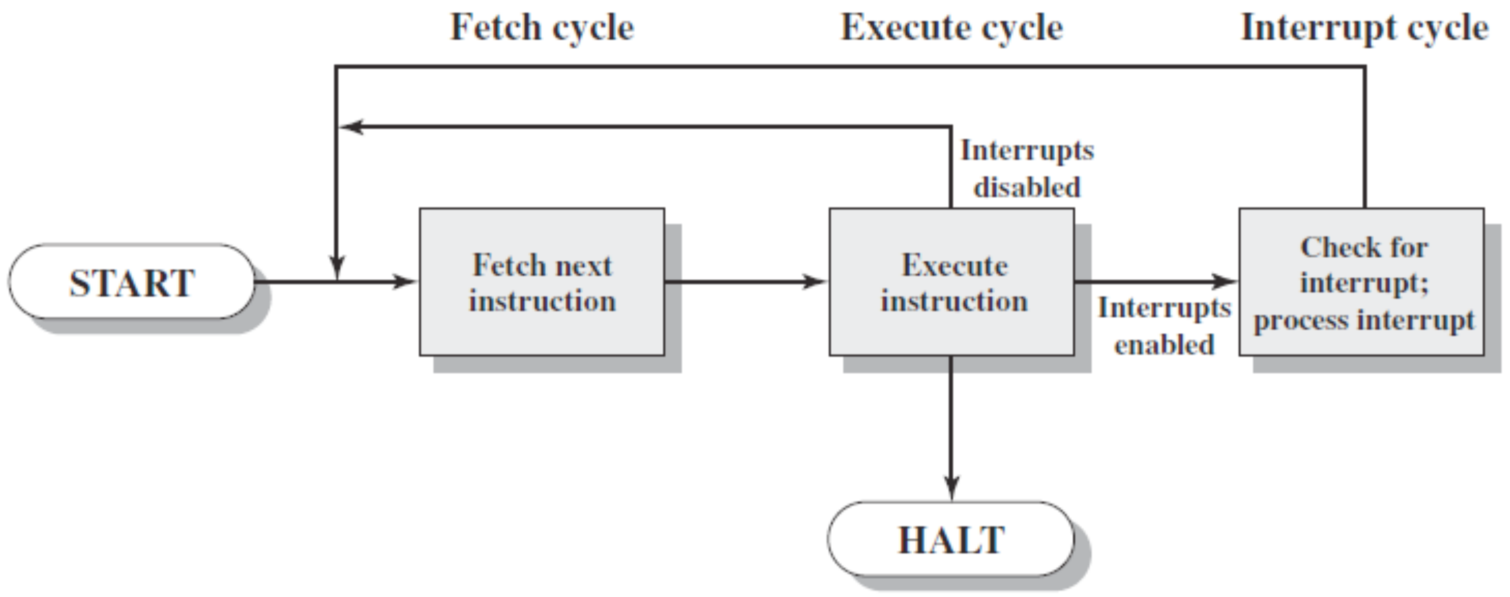


Figure 2. Instruction Cycle with Interrupts

Instruction Cycle with Interrupts

To accommodate interrupts, an interrupt cycle is added to the instruction cycle.

The processor checks to see if any interrupts have occurred, indicated by the presence of an interrupt signal.

If no interrupts are pending, the processor proceeds to the fetch cycle and fetches the next instruction on the current program.

Instruction Cycle with Interrupts

If an interrupt is pending, the processor does the following:

- It suspends execution of the current program being executed and saves its context. This means saving the address of the next instruction to be executed (current contents of the program counter) and any other data relevant to the processor's current activity

- The processor proceeds to the fetch cycle and fetches the the first instruction in the interrupt handler program. The interrupt handler program is part of the OS.

Pipelining

An instruction has a number of stages or tasks that occur in sequence.

Pipeline: chain of instructions where the output of one unit is the input of the next unit.

The pipeline has two independent stages. The first stage fetches an instruction and buffers it. When the second stage is free , the first stage passes it the buffered instruction.

While the second stage is executing, the first stage takes advantage of any unused memory cycles to fetch and buffer the next instruction. This is called instruction prefetch or fetch overlap.

Pipelining

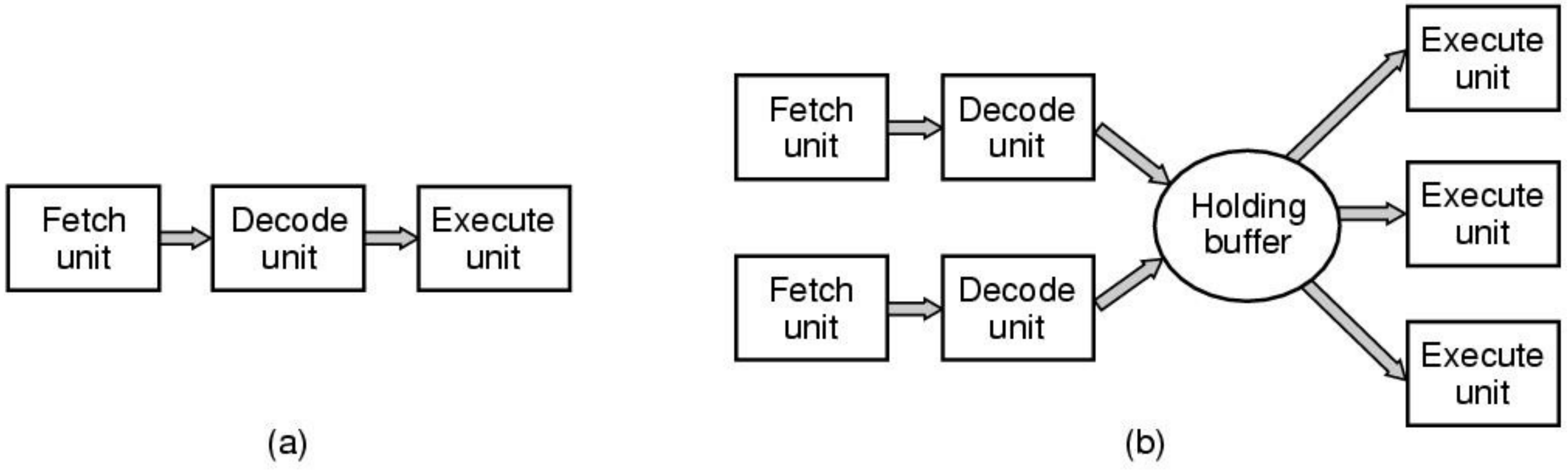


Figure 3. (a) A three-stage pipeline. (b) A superscalar CPU.

I/O Resource Management

The operating system is responsible for:

- Issuing commands to I/O devices
- Handling all interrupts and errors generated by the devices.
- The operating system needs some way to efficiently manage all of these devices and the flow of data coming in from them, or going out to them.
- These responsibilities are complicated by the fact that many processes being executed by the operating system may need to share the same I/O resources.

What are I/O Resources?

- Input/Output resources are any I/O devices (and their supporting hardware and software components) that are available for use by processes.

- Resources are frequently shared between processes
  - The operating system must have some way to regulate access to the resources to prevent conflicts and deadlocks.

Device Controllers

- Device controllers are components on the motherboard (or on expansion cards) that act as an interface between the CPU and the actual device.
- Device controllers interpret the instructions by comparing them a list of device commands stored on the controller, which then forwards the appropriate command directly to the device.
- When a device needs to access the CPU, the device controller issues an **Interrupt Request (IRQ),**

Common Device Controllers

- Keyboard Controller – controls the keyboard and PS/2 mouse (not always needed in newer systems)
- DMA Controller – controls Direct Memory Access
- Network Adaptor Controller – controls the Network Adaptor/ Network Interface Card (NIC)
- IDE Controller – controls EIDE devices, including the hard disk and CD/DVD drive
- Graphics Adaptor – controls video output devices, such as a monitor or LCD projector
- USB Controller – controls devices connected by USB

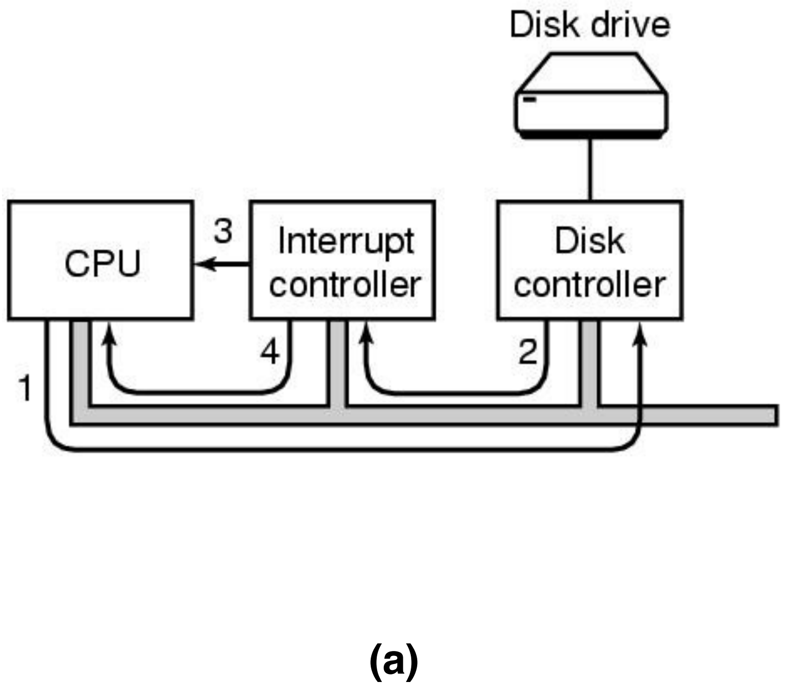
The Interrupt Controller

- The interrupt controller is a special component on the motherboard that:
  - manages all interrupts,
  - prioritizes them based on a predetermined priority sequence, and
  - forwards the interrupts to the CPU.

The Interrupt Controller

- When a device wants the CPU’s attention, its device driver initiates an interrupt request (IRQ).
- Each type of device has a different IRQ number assigned to it.
- When multiple devices signal for the CPU’s attention at the same time, the interrupt controller checks their IRQ number, and places them in a queue.
- The device with the lowest IRQ number gets the highest priority

I/O Devices



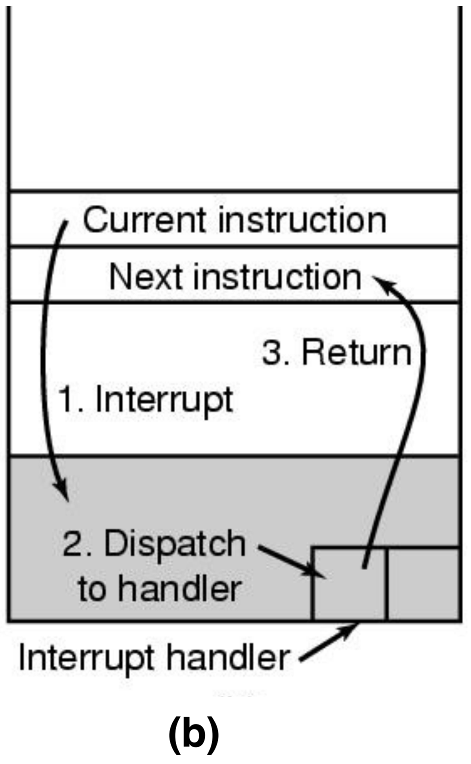


Figure 4. (a) The steps in starting an I/O device and getting an interrupt.

Programmed I/O

Data is exchanged between the processor and the I/O Module.

The processor executes a program that gives it direct control of the I/O operation, including sensing device status, sending a read or write command, and transferring data.

When the processor issues a command to the I/O module, it must wait until the I/O operation is complete.

If the processor is faster than the I/O module, this is wasteful of processor time.

Interrupt-Driven I/O

The processor issues an I/O command, continues to execute other instructions, and is interrupted by the I/O module when the latter has completed its work.

With programmed and interrupt I/O, the processor is responsible for extracting data from main memory for output and storing the data in main memory for input.

Direct Memory Access (DMA)

The alternative to programmed and interrupt I/O is Direct Memory Access (DMA). The I/O module and main memory exchange data directly without processor involvement.

Involves an additional module on the system bus. The DMA module is capable of mimicking the processor and of taking control of the system from the processor.

It needs to do this to transfer data to and from memory over the system bus. The DMA must use the bus only when the processor does not need it, or it must force the processor to suspend operations.

Direct Memory Access (DMA)

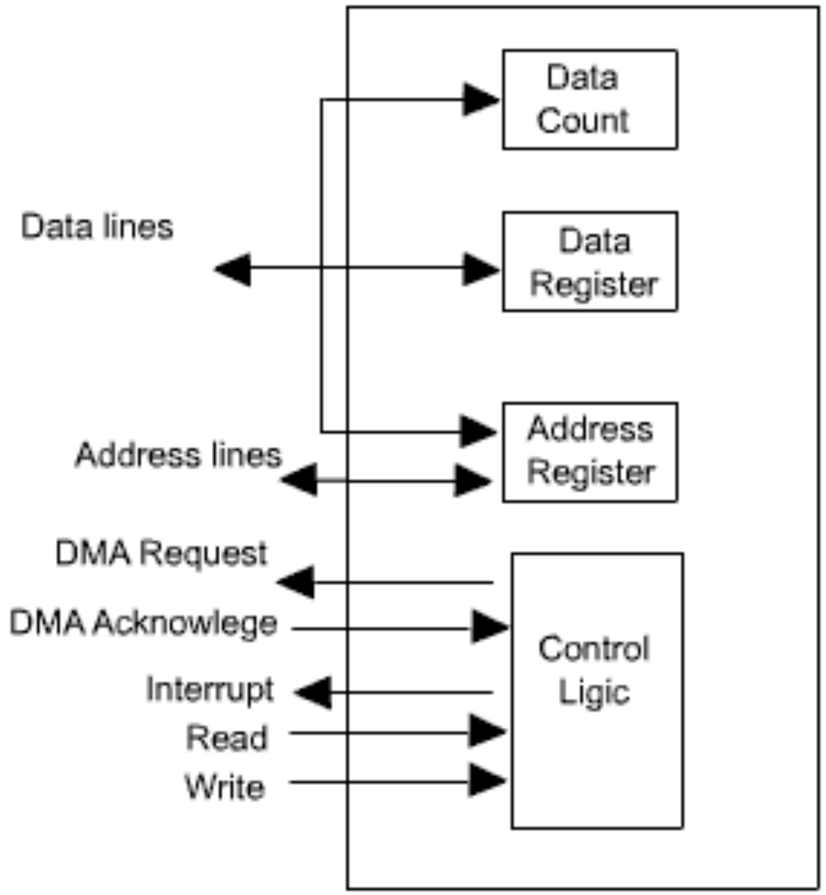


Figure 5. Typical DMA Block Diagram

Instruction Set Architectures

An instruction set architecture (ISA) is a well-defined interface between the hardware and software of a computer.

An ISA can be considered as a “contract” between software and hardware.

- Functional definition of operations, modes, and storage locations supported by hardware.
- Precise description of how to invoke, and access them

No guarantees regarding

- How operations are implemented
- Which operations are fast and which are slow
- Which operations take more power and which take less

Instruction Set Architectures

An ISA permits multiple implementations that may vary in performance, physical size, and monetary cost (among other things); because the ISA serves as the interface between software and hardware.

CISC Architectures

CISC (Complete Instruction Set Computer) architecture

Motivation:

- Support for high-level programming languages
- Simplify compilers (larger number of instructions, more complex instructions)
- Improve performance

RISC Architecture

RISC (Reduced Instruction Set Computer) architecture

Characteristics:

- One instruction per cycle
- Register-to-register operations
- Simple addressing modes
- Simple instruction formats

x86 Architecture

- Based on the Intel 8086 CPU and its Intel 8088 variant.
- The majority of personal computers and laptops sold (up to 2017) are based on the x86 architecture.
- Also used in midrange computers, workstations, servers and most new supercomputer clusters of the TOP500 list.
- A large amount of software, including operating systems (OSs) such as DOS, Windows, Linux, FreeBSD, NetBSD, OpenBSD, Solaris and macOS, functions with x86-based hardware.
- The x86 architecture is a variable instruction length, primarily "CISC" design with emphasis on backward compatibility.

x86_64

- x86-64 (also known as x64, x86_64, AMD64 and Intel 64) is the 64-bit version of the x86 instruction set.
- It supports vastly larger amounts (theoretically, 264 bytes or 16 exabytes) of virtual memory and physical memory than is possible on its 32-bit predecessor
- Allows programs to store larger amounts of data in memory.
- The original specification, created by AMD and released in 2000, has been implemented by AMD, Intel and VIA.

IA-64

- The instruction set architecture (ISA) of the Itanium family of 64-bit Intel microprocessors (ISA spec originated at Hewlett-Packard (HP))
- It is based on explicit instruction-level parallelism, in which the compiler decides which instructions to execute in parallel.
- Contrasts with other superscalar architectures, which depend on the processor to manage instruction dependencies at runtime.
- Concern: reduced instruction set computing (RISC) architectures were approaching a processing limit at one instruction per cycle.
- Supports hardware multithreading: Each processor core maintains context for two threads of execution.
- Hardware support for virtualization

ARM

- Originally Acorn RISC Machine, later Advanced RISC Machine. - ARM Holdings, Cambridge UK

- Based on RISC architecture typically require fewer transistors than those with a complex instruction set computing (CISC) architecture (such as the x86 processors).

- Improves cost, power consumption, and heat dissipation. These characteristics are desirable for light, portable, battery-powered devices.

- With over 100 billion ARM processors produced as of 2017, ARM is the most widely used instruction set architecture in terms of quantity produced.

- ARMv8-A architecture adds support for a 64-bit address space and 64-bit arithmetic with its new instruction set.

- Supported by a large number of embedded and real-time operating systems,

SPARC

- SPARC - Scalable Processor Architecture. RISC instruction set architecture (ISA) originally developed by Sun Microsystems, Santa Clara , CA, USA

- Heavily influenced by the earlier RISC designs. Minimalist, including as few features or op-codes as possible and aiming to execute instructions at a rate of almost one instruction per clock cycle.

- Successful early commercial RISC system. Oracle killed off SPARC design in 2017.

- OS supported: SunOS, Solaris, OpenSolaris, FreeBSD, OpenBSD, NetBSD, and Linux