CSc33200: Operating Systems, CS-CCNY, Fall 2003 Jinzhong Niu September 17, 2003

Computer Hardware Basics

1 Questions arising from No. 2 assignment

- What are the meanings of the following terms?
  1. 32-bit microprocessor, and 32-bit instructions: see section 2.1.
  2. opcode, immediate operand, and operand address: see section 2.2.
  3. local data bus, and local address bus: see section 2.3.
  4. machine cycle, processor cycle, and bus cycle: see section 2.4.
- In problem 2, when programmed I/O is used, since I/O devices are usually much slower than the processor, is it proper to discuss I/O transfer rate without considering the speed of I/O device?

2 Terms

2.1 8-bit, 16-bit, 32-bit

We often see terms like *32-bit processor*, *64-bit operating system*, *16-bit system bus*? What does the number exactly mean? *Generally, it refers to the number of bits that can be processed or transmitted in parallel, or the number of bits used for single element in a data format.*

The entities that are usually modified by these numbers are:

- **Bus**: The number, what we call the *size* or *width* of bus, indicates the number of wires in the bus. A 32-bit bus transmits 32 bits in parallel.
- **Microprocessor**: This adjective refers to the number of bits used internally by a computer's CPU, or concretely the width (in bits) of the processor's general-purpose registers, which determine how much data it can compute with at a time. The term is also however often misused, and sometimes people refer to the size of a processor based on its data bus width (for example) which isn't really correct. Interestingly, every processor introduced

in the last decade, from the earliest 386SX to the latest Pentium II, is a 32 bit processor, based on this definition. So this isn't going to be something you use to differentiate between CPUs.

- **Software**: A number is also used for software, which means the software has been developed assuming the underlying hardware platform (CPU) has the corresponding size, and it is the number of bits used to represent memory addresses. For example, in Intel's *segmented space model*, 16-bit applications will be compiled into segments of data and/or code, the length of each not exceeding 64k since 16-bit processors can only manipulate a segment shorter than 64K. As the more functions are needed from a single piece of software, the larger it has to be, the size of microprocessor will increase in some certain period to meet the application requirements. Sometimes the upgrade may be a marketing strategy of companies.

- **Operating system**: When these numbers modify operating systems, we refer to those OSes that are developed to run on the corresponding hardware platform and provide related support for application programs, e.g. compiling program source code into segments of code with proper segment limit, in which memory addresses are of the corresponding width.

The number indicates the length of integers that a processor (or concretely ALU) can compute on at once. The bus among processor, memory, and I/O modules usually has the same width, but not always. Table 1 shows the differences among members of Intel x86 microprocessor family in this aspect:

Generations	1		2	3	4	5
Model	8088	8086	80286	80386DX	80486	Pentium
Data bus width	8	16	16	32	32	64
Address Bus Width	20	20	24	32	32	32
Register size	16	16	16	32	32	32

Table 1: x86 microprocessors

The first chip used in PCs was Intel's 8088. This was not, at the time it was chosen, the best available CPU, in fact Intel's own 8086 was more powerful and had been released earlier. The 8088 was chosen for reasons of economics: its 8-bit data bus required less costly motherboards than the 16-bit 8086. First generation processors were used on the earliest of machines, the original IBM PC and XT, and the first clones. These machines were primitive in most ways compared to modern PCs, and the processors they used were of limited capabilities. First generation processors run at system bus speed and use the oldest processor technologies. On 8088, a program may consist of several segments whose length are no more 64k. At those time, a whole operating system, say DOS, can be contained within a floppy disk.

80286 is generally considered the only chip in the second generation. Intel also made a chip between the 8088 and the 80286, called the 80186. Largely due to a lack of performance enhancements and compatibility problems with support chips, the 80186 was never really used in PCs. The 286 was the first major step up in PC processors, providing significant performance increases over the 8088 and 8086 - double or more performance at the same clock speed. The 286 also widened the address bus to allow access to 16 MB of memory, and introduced protected mode operation. At this time DOS was still the virtually exclusive operating system, and the protected mode was largely ignored.

The third generation processors (the 386 family) represent another step forward from the second generation class. Intel started with this generation of chips to create ”subfamilies” of related chips with different capabilities, using the ”DX” and ”SX” designations. The Intel 80386DX was the first true 32-bit processor used on the PC platform. Its internal register size was increased to 32 bits, and its data and address buses were as well, increasing addressable memory to 4 GB theoretically. The 386DX's increased power and the improved processor modes it offered (including full protected mode and virtual real modes) spurred the introduction of GUI-based operating systems on the PC, such as Microsoft Windows.

The Intel 80486DX was the first member of the 486 family. It provided a very significant increase in power over the 80386DX processor, in fact, far more proportionately than the 386 did over the 286. A 486DX processor provides approximately 100 to 150% more performance than a 386DX of the same clock speed. The 80486 brought GUIs to the mainstream on PCs; it is the minimum processor that most people consider ”usable” for running an operating system like Microsoft Windows.

Interestingly, the 486 does not provide its performance improvements by widening any of the buses, as had been the case in the previous two generations: it is still a 32-bit processor with 32-bit data and address buses, just like the 386DX. However, internally, the 486 incorporates several significant improvements over the 386:

- **Deeper Pipeline**: The execution pipeline was increased by one step.
- **Primary Cache**: The 486 processor was the first to incorporate level 1 cache on the chip, to reduce the number of required accesses to main memory.
- **Integrated Floating Point Unit**: The chip includes an integrated math coprocessor (not on the SX version however). In addition, the coprocessor provides much more performance than the optional 80387 used with 386 chips, in part because it is integrated into the chip.

Intel's new fifth-generation chip was expected to be called the 586, following their earlier naming conventions. However, with the rise of AMD and Cyrix, Intel wanted to be able to register as a trademark the name of their new CPU, and numbers can't be trademarked. Thus, the Pentium was born.

The Pentium provides greatly increased performance over the 486 chips that precede it, due to several architectural changes. Roughly speaking, a Pentium chip is double the speed of a 486 chip of the same clock speed.

We won't go any further along the development of Intel processor family, for more information, please refer to related materials.

2.2 Opcode, operand, and addressing mode

An machine instruction is made up of an **Opcode** and one or more **operands**. The opcode tells the processor what action to perform over the following operands.

The methods for specify the operand(s) for a machine instruction are called **addressing modes**. Different processors vary greatly in the number of addressing modes they provide. The more complex modes described below can usually be replaced with a short sequence of instructions using only simpler modes.

The most common modes are "*register*" - the operand is stored in a specified register; "*absolute*" - the operand is stored at a specified memory address; and "*immediate*" - the operand is contained within the instruction.

Most processors also have *indirect addressing modes*, e.g. "register indirect", "memory indirect" where the specified register or memory location does not contain the operand but contains its address, known as the "*effective address*". For an absolute addressing mode, the effective address is contained within the instruction.

Indirect addressing modes often have options for *pre- or post- increment or decrement*, meaning that the register or memory location containing the effective address is incremented or decremented by some amount (either fixed or also specified in the instruction), either before or after the instruction is executed. These are very useful for stacks and for accessing blocks of data. Other variations form the effective address by adding together one or more registers and one or more constants which may themselves be direct or indirect. Such complex addressing modes are designed to support access to multidimensional arrays and arrays of data structures.

The addressing mode may be "*implicit*" - the location of the operand is obvious from the particular instruction. This would be the case for an instruction that modified a particular control register in the CPU or, in a stack based processor where operands are always on the top of the stack.

2.3 Bus

A bus is a collection of wires through which data is transmitted from one part of a computer to another. You can think of a bus as a highway on which data travels within a computer. When used in reference to personal computers, the term bus usually refers to *internal bus*, which connects all the internal computer components to the CPU and main memory.

According to the type of data transmitted through the bus, we have:

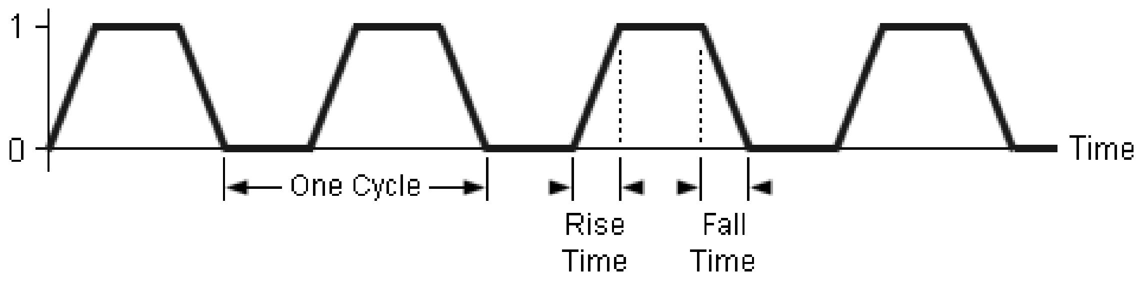
- **Data bus**: The bus used to carry actual data.
- **Address bus**: The bus transferring information about where the data should go.
- **Control bus**: The physical connections that carry control information between the CPU and other devices within the computer. Whereas the data bus carries actual data that is being processed, the control bus carries signals that report the status of various devices, or ask devices to take specific actions. For example, one line of the bus is used to indicate whether the CPU is currently reading from or writing to main memory.

According to the location of buses, or what kinds of components they connect, we have:

- **Local bus**: Another name of *system bus*, which connects the microprocessor, main memory, and all kinds of I/O modules. Although local buses can support only a few devices, they provide very fast throughput. Modern PCs include both a PCI local bus and a more general ISA expansion bus for devices that do not require such fast data throughput. A local bus is also called *frontside bus*, *memory bus*, or *host bus.*
- **backside bus**: A microprocessor bus that connects the CPU to a Level 2 cache. Typically, a backside bus runs at a faster clock speed than the frontside bus that connects the CPU to main memory. For example, the Pentium Pro microprocessor actually consists of two chips – one contains the CPU and the primary cache, and the second contains the secondary cache. A backside bus connects the two chips at the same clock rate as the CPU itself (at least 200 MHz). In contrast, the frontside bus runs at only a fraction of the CPU clock speed.
- **External bus**: A bus that connects a computer to peripheral devices. Two examples are the Universal Serial Bus (USB) and IEEE 1394.

2.4 Speeds

- **System clock** There are so many circuits inside a computer. To perform a specific action, it is necessary for some sort of synchronization to occur so that different parts function and cooperate as expected. The "conductor" of the PC is the *system clock*, which brings synchronization. A clock is just a signal that alternates between zero and one, back and forth, at a specific pace. In many ways, it is just like a metronome, going back and forth over and over. The hero behind the clock is a crystal that can oscillate at a constant rate. The clock sets the "pace" for everything that happens within a particular electronic circuit. The pulses generated by the clock is illustrated by the following figure.



A **clock cycle** refers to a single complete traversal of the signal, from the rising edge, through the time when the value of the clock is one, through the falling edge, the time that the value is zero, until the start of the next rising edge. (You can actually "chop" the signal wherever you want and have it be a single cycle, as long as you only cover one cycle without "overlapping").

The **clock rate** or **clock speed** is simply the reciprocal of the cycle time, and is therefore the number of cycles that occur each second (as opposed to the number of seconds per cycle). The original IBM PC had a clock rate of 4.77 MHz (almost five million cycles/second). As of 1995, Intel's Pentium chip runs at 100 MHz (100 million cycles/second). The clock rate of a computer is only useful for providing comparisons between computer chips in the same processor family. An IBM PC with an Intel 486 CPU running at 50 MHz will be about twice as fast as one with the same CPU, memory and display running at 25 MHz. However, there are many other factors to consider when comparing different computers. Clock rate should not be used when comparing different computers or different processor families. For example, it makes no sense to compare the performance of a Pentium PC system with a Power-PC macintosh system merely based on their clock rates. Rather, some benchmark should be used. Clock rate can be very misleading even when a same processor family is concerned, since the amount of work different computer chips can do in one cycle varies.

The earliest PCs had just a single system clock; everything from the CPU and the memory to the system bus and peripherals ran at the same speed. Today, PC components are much more specialized, and some circuits and components operate much faster than others. For this reason, there is not just one system clock within the PC, but several. The common facilities to achieve this goal are **frequency multipliers** and **frequency dividers**. For example, a frequency multiplier could take a 100 MHz clock and create from it a 200 MHz clock signal. Some computers provide jumpers on the motherboard so that users may alter the parameter of a frequency multipliers, i.e. how many times a clock should be generated from the system clock.

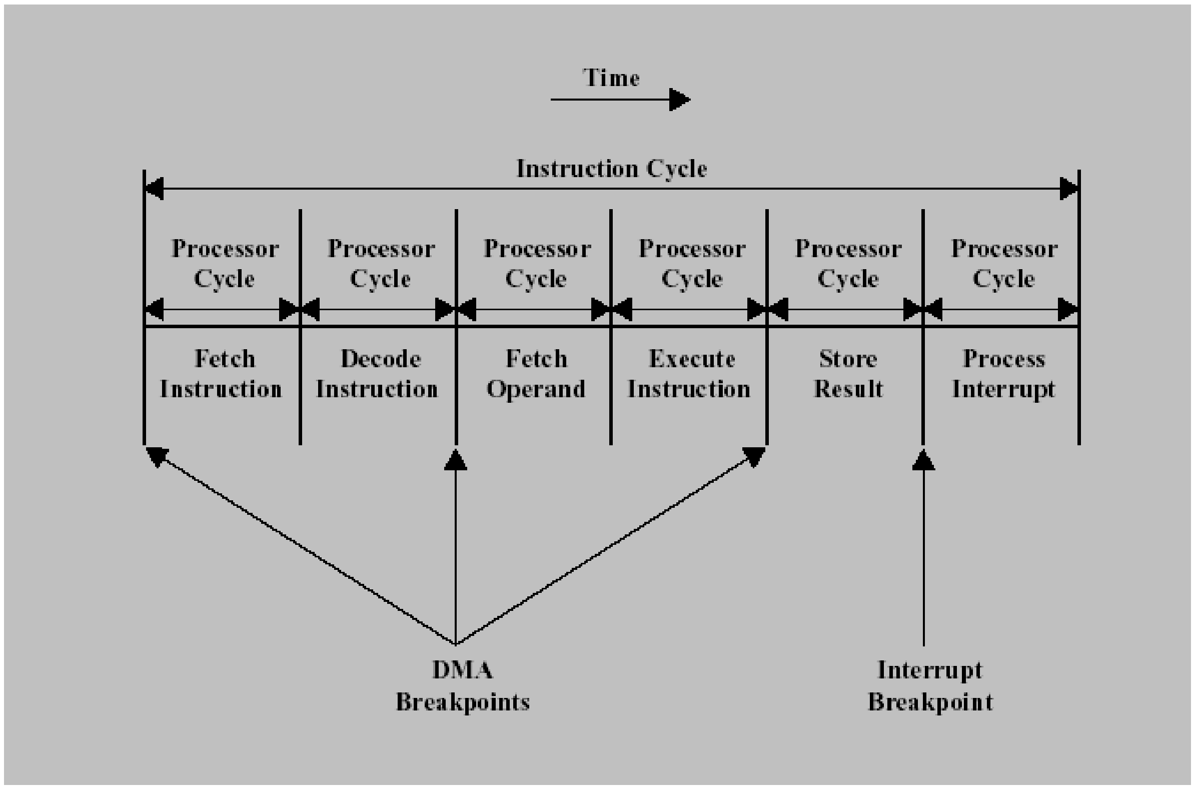
- **Memory Access time**: also known as response time or latency, refers to how quickly the memory can respond to a read or write request.
- **Memory cycle**: refers to the minimum period between two successive requests. For various reasons the time separating two successive requests is not always 0, i.e a memory with a response time of 80 ns cannot satisfy a request every 80 ns.
- **Machine cycle**: also called *instruction cycle*, the four steps which the CPU carries out for each machine language instruction: *fetch*, *decode*, *execute*, and *store*. These steps are performed by the control unit, and may be fixed in the logic of the CPU or may be programmed as microcode which is itself usually fixed (in ROM) but may be (partially) modifiable (stored in RAM).

  The fetch cycle places the current program counter contents (the address of the next instruction to execute) on the address bus and reads in the word at that location into the instruction register (IR).

  The decode cycle uses the contents of the IR to determine which gates should be opened between the CPU's various functional units and buses and what operation the ALU(s) should perform (e.g. add, bitwise and). Each gate allows data to flow from one unit to another (e.g. from register 0 to ALU input 1) or enables data from one output onto a certain bus. In the simplest case (”horizontal encoding”) each bit of the instruction register controls a single gate or several bits may control the ALU operation. This is rarely used because it requires long instruction words (such an architecture is sometimes called a very long instruction word architecture). Commonly, groups of bits from the IR are fed through decoders to control higher level aspects of the CPU's operation, e.g. source and destination registers, addressing mode and ALU operation. This is known as vertical encoding. One way *RISC*, Reduced Instruction-Set Computer, processors gain their advantage in speed is by having simple instruction decoding which can be performed quickly.

  The execute cycle occurs when the decoding logic has settled and entails the passing of values between the various function units and buses and the operation of the ALU. A simple instruction will require only a single execute cycle whereas a complex instruction (e.g. one using memory indirect addressing) may require three or four. Instructions in a RISC typically (but not invariably) take only a single cycle.

The store cycle is when the result of the instruction is written to its destination, either a register or a memory location. This is really part of the execute cycle because some instructions may write to multiple destinations as part of their execution.



2.5 Caches

- **Level 1 (Primary) Cache**: the fastest memory on the PC. It is in fact, built directly into the processor itself. This cache is very small, generally from 8 KB to 64 KB, but it is extremely fast; it runs at the same speed as the processor. If the processor requests information and can find it in the level 1 cache, that is the best case, because the information is there immediately and the system does not have to wait.

  Level 1 cache is also sometimes called *internal cache* since it resides within the processor.

- **Level 2 (Secondary) Cache**: The level 2 cache is a secondary cache to the level 1 cache, and is larger and slightly slower. It is used to catch recent accesses that are not caught by the level 1 cache, and is usually 64 KB to 2 MB in size. Level 2 cache is usually found either on the motherboard or a daughterboard that inserts into the motherboard. Pentium Pro processors actually have the level 2 cache in the same package as the processor itself (though it isn't in the same circuit where the processor and level 1 cache are) which means it runs much faster than level 2 cache that is separate and resides on the motherboard. Pentium II processors are in the middle; their cache runs at half the speed of the CPU.

Level 2 cache is also sometimes called *external cache* since it resides outside the processor.