**Schaum’s outlines**

**Principles of Computer Science**

**Computer Organization**

Earlier computers were wired for a certain function (e.g. early calculators). The program was built into the construction of the computer.

The main idea of the von Neumann architecture (1945) is that the program to be executed resides in the computer’s memory, along with the program’s data.

The von Neumann architecture is also called the “stored program computer”.

\**10’s complement for the decimal is like 2’s complement for the binary.*

**About ASCII**

In ASCII each character is assigned a 7-bit integer value. For instance, ‘A’ = 65 = 1**0**00001, ‘B’ = 66 = 100010, etc.

Lowercase letters are assigned a different set of numbers: ‘a’ = 97 = 1100001, ‘b’ = 98 = 1**1**00010, etc.

Other characters, like $, &, etc. are also introduced, as well as, **control characters** (do not print, but can be used in streams of characters to control devices). ‘linefeed’ means \n — new line.

Unicode is the most popular way of character encoding, and it is backwards compatible with ASCII.

How does a program know, whether to interpret the sequence of bits as an integer, or, let’s say a character? The program will interpret the bits based on its expectation.

**CPU/ALU**

Both the control unit and the ALU include special, very high-performance memory cells called **registers.** Registers are intimately connected to the wiring of the control unit and the ALU. Some have special purpose, and some are general purpose.

**FROM THE QUIZ:** Access to registers is faster than access to main memory.

The number of bits required to specify a register is fewer than to reference a main memory address. Therefore, instructions can be smaller in size and faster to load. Why less bits are required? Because main memory will be very large, so more bits will be required to specify a memory address.

Program Counter (PC) is the example of a special-purpose register.

The PC keeps track of the address of the instruction to execute next.

When the control unit begins Fetch/Execute cycle, the control unit moves the instruction stored at the address saved in the PC to another special register called the **instruction register (IR).**

When the fetch of the next instruction occurs, the control unit automatically increments the PC, so that the PC now “points” to the next instructions in sequence.

The control unit decodes the instruction in the IR.

Other registers of the ALU are general purpose. They are used to store data close to the processor, where the processor can access the information even more quickly than the value is in the memory. Different computers have different number of registers, and the size of the registers will be congruent with the word size of the computer (16-bit, 32-bit, etc.).

The number of registers, and the nature of the special-purpose registers, comprise an important part of the computer architecture.

congruent with — совпадать с

**Instruction set**

The instruction set is rather limited, and depends on the machine currently used.

Each machine has its own instruction set. A computer’s assembly language corresponds to its instruction set.

The IBM 360 family of computers was the first to introduce a shared instruction set for different computers within the family. It was a breakthrough at the time.

When the program is written in a high-level language, the language processor translates your code to the appropriate language processor on the new computer. The source code may not change, but the translation of this code into machine instructions will be different, because of the difference in the instruction set of a given machine.

**The language processor** has the responsibility to translate standard higher-level programming syntax into the correct machine instruction bit patterns.

Machine instructions are represented in a computer word as 1s and 0s.

**“Op-code”** is the part of the computer word that stores the operation to perform. E.g. ADD, Jump, Compare, etc.

Other bits in the instruction word specify the values to operate on, the “operands”. An operand might be a register, a memory location, or a value already in the instruction word operand field.

In the Intel architecture, instructions (“code”) are stored in a separate section of memory from data. When the computer fetched the next instruction, it does so from the code section of memory.

JMP instruction:

11101001 11111100 01001111

The first byte is the op-code for JMP direct (the address provided is where we want to go, not a memory location holding the address to which we want to go).

The second byte is the *low-order byte* for the address to which to jump. The third byte is *the high-order byte* for the address to which to jump. This means that the proper address looks like this:

01001111 11111100

Intel processors historically are “little endian”. It stores the least significant byte of a multiple byte value at the lower (first) address.

Other computers, like the IBM 370, or **the MIPS,** are “big endian”, so the most significant byte is stored first.

There is no performance reason to prefer big endian or little endian formats. The formats are a product of history.

Today, big endian order is the standard for network data transfers, but only because the original TCP/IP Protocols were developed on big endian machines.

In general, instructions fall into these categories: data transfer, input/output, arithmetic operations, logical operations, control transfer, and comparison.

**Memory**

Memory is used to store program instructions and data. The basic operations on memory are store and retrieve.

Word size represents the number of bits the computer usually processes at one time.

***Computer word size***

*Each computer deals with a certain number of bits at a time. Early hobbyist computers manipulated 8 bits at a time, and so were called “8-bit computers”. That was the computer word size. The computer might be programmed to operate on more than 8 bits, but it’s basic operations dealt with 8 bits at a time. Nowadays, the word size is either 32 or 64 bits.*

Today byte is the measure of computer memory, and most computers, regardless of word size, offer “byte addressability”, meaning that each byte has a unique memory address. Even though the computer may be a 32-bit machine, each byte in the 4-byte computer word (32 bits) can be addressed uniquely, and it’s value can be read and updated.

There are at least two registers associated with the memory control circuitry to facilitate storage and retrieval. These are the **memory address register (MAR)** and the **memory data register (MDR).**

When writing to memory, the CPU first transfers the value to be written to the MDR, and the address of the location to be used to the MAR. At the next memory access cycle, the value in the MDR will be copied into the location identified by the contents of the MAR.

When retrieving from memory, the CPU first stores the address to read in the MAR. When the read occurs on the next memory access cycle, the value in that location is copied into the MDR. From the MDR in the memory controller, the data value can be transferred to one of the registers or elsewhere.

The opposite of random access is serial access. E.g. the magnetic tape.

**Cache memory** is small, high-performance memory located close to the CPU (or even on the same electronic chip as the CPU). It is used to hold a copy of the contents of a small number of main memory locations.

It is useful, because program execution demonstrates a property called “localily of reference”.

**Locality of reference** — for relatively long periods of time, the execution of a program will reference and affect a small number of memory locations. Accesses to memory are not random.

So, when computer copies the contents of main memory currently being accessed to cache memory, the CPU can avoid waiting for access to slower main memory, and access the cache instead. Access to cache typically 5 to 10 times faster than access times for main memory.

**Cache coherency** means ensuring that the cache memory is synchronized with the contents of the main memory. Also, cache management algorithm is used.

——————

\*How to prove that the 2’s complement actually represents the negatively signed number.

1. E.g -6 in 8 bits
2. In binary it’s 0000 0110
3. Flip the bits: 1111 1001
4. Add 1: 1111 1010

If we add 1 six times, we will get 0 as the result. That’s how we can check, whether it’s true. Also, we can do calculations, and see, whether we get the right answer in both, binary and decimal.