3. Programming

Frank Stephan

January 23, 2014

Carrot Cake Recipe (adapted from allrecipes.com)

1. Preheat the oven to 175 degrees Celsius.
2. Grease and flour a 9 inches by 13 inches pan.
3. Beat four eggs.
4. Mix the eggs, one fourth of a cup of vegetable oil, two cups of sugar, two cups of flour, two tea spoons of backing soda, and three cups of grated carrots in a large bowl.
5. Add two tea spoons of vanilla extract if you have some.
6. Pour the mixture into the pan.
7. Bake until a knife inserted into the center comes out clean.
8. Let cool for ten minutes.
9. Put in a plate.
10. Prepare the frosting.
11. Pour the frosting onto the cake.

Von Neumann Architecture

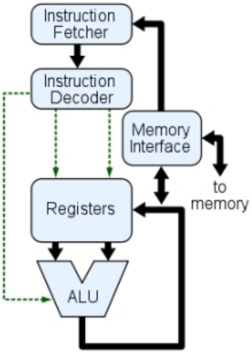
The Von Neumann architecture was proposed in 1945 by the mathematician and computer scientist John von Neumann.

Von Neumann Architecture

The Von Neumann Architecture describes a computer as consisting of a processing unit with an arithmetic logic unit and registers, a control unit containing an instruction register and program counter, a memory to store both data and instructions, external mass storage, and input and output mechanisms.

Central Processing Unit

The central processing unit (CPU) is composed of the arithmetic logic unit (ALU), registers, an interface to main memory and an instruction fetcher and decoder.



Central Processing Unit

The Central Processing Unit repeatedly fetches, decodes and executes program instructions, and writebacks results.
The program is in memory. It is a series of instructions each represented by one binary word.

1. The CPU fetches the next instruction at a location in memory determined by the program counter.
2. The CPU decodes the instruction and determines what it has to do as defined by the CPU's instruction set architecture (ISA).
3. The ALU performs the arithmetic and logical operation required on the inputs (in the registers or main memory).
4. The CPU writes the results to memory (register or main memory).

Instruction Set

Every processor or processor family has its own machine code instruction set.

Instructions

move, add, substract, multiply, divide, increment, decrement, exchange, compare, jump on condition, etc.

x86 Instruction Format

x86 instructions are represented as binary numbers and require between 1 and 6 bytes. Most instruction are coded on 2 bytes (16 bits) as follows.

- 6 bits for the code of the operation
- 1 bit for the direction of data movement (1 for movement from second to first operand, 0 otherwise)
- 1 bit for the size of the operands (1 for word - 16 bits or 32 bits machine - and 0 for byte)
- 2 bits for the interpretation of the second operand
- 3 bits for the code of the first operand (a register)
- 3 bits for the code or address of the second operand (a register or memory)

Example

The instruction 100010 1 1 00 000 111 (8B 07 in hexadecimal)) moves the value at the address in the register bx into the register ax.

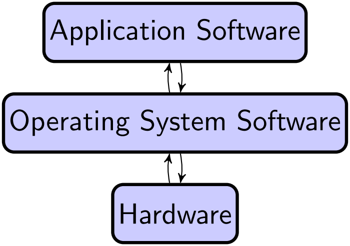
Assembly Language

The machine is programmed in assembly language. An assembler generates the machine code.

```
mov ax, [bx]
```

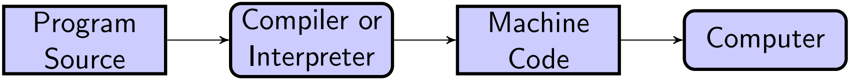
```
.data
        Sum    DW ?      ; non-initialised 2-byte value.
        Length DW 6      ;     initialised 2-byte value.
        Table  DB 89, 53, 5, 61, 127, 5
          ; Table is an array of Length 1-byte elements.

.code
        lea bx, Table       ; bx receives Table's address.
        mov ax, 0           ; let ax take the value 0.
        mov si, 0           ; let si take the value 0.
next:
        cmp si, Length      ; compare si with Length (6).
        je finish           ; if equal, go to "finish".
        add ax, byte ptr [bx+si]
          ; add to ax the 1-byte value pointed by bx+si.
        inc si              ; add to si the value 1.
        jmp next            ; go to "next".
finish:
        mov Sum, ax         ; store the result into Sum.
```



The Operating System

The operating system software (e.g. Android, iOS, GNU/Linux, Mac OS X, Unix BSD, Microsoft Windows) provides the interface to the computer hardware (central processing unit and main memory) and devices (e.g. keyboard, screen, printer, hard drive, DVD, network cards etc. by means of drivers) and generic services and abstractions (such as memory management, multitasking, multiprocessing and file and directory management) for the application software. It often includes a user interface.



Programming Languages

The application is written in a higher level programming language.

Compilers and Interpreters

- The program source written in a programming language is compiled or interpreted by a compiler or interpreter, respectively, and executed on the hardware with the mediation of the operating system.

Some Programming and other Languages

ABAP ACSL Ada Algol Ant APL Assembler Awk bash Basic C C++ Caml Clean Cobol Comal csh Delphi Eiffel Elan erlang Euphoria Fortran GCL Gnuplot Haskell HTML IDL inform Java JVMIS ksh Lisp Logo make Mathematica Matlab Mercury MetaPost Miranda Mizar ML Modelica Modula-2 MuPAD NASTRAN Oberon-2 OCL Octave Oz Pascal Perl PHP PL/I Plasm POV Prolog Promela Python R Reduce Rexx RSL Ruby S SAS Scheme Scilab sh SHELXL Simula SmallTalk SQL tcl TeX VBScript Verilog VHDL VRML XML XSLT

Data

A programming language provides variables, data structures and other objects to record, organize and access values of various data types.

- Numbers (integer, floating point real),
- Strings,
- Boolean,
- etc.

Operations

A programming language provides constructs, operations and functions to create and manipulate the data, data structures, objects and to access resources.

- Assignment,
- Arithmetic operations,
- Boolean operations and conditions,
- Input/Output operations,
- etc.

Control Structures

A programming language provides control structures to define the execution flow at runtime.

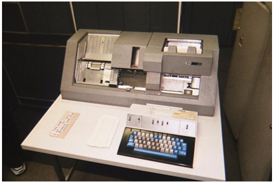
- Direct sequencing,
- Jump,
- Conditional branching,
- Bounded iteration,
- Conditional iteration,
- Subroutines,
- Recursion.

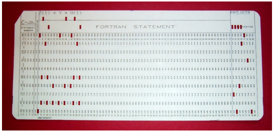
FORTRAN

Fortran is an imperative programming language designed for scientific computing by IBM in the 1950s.

```
PROGRAM MAIN
INTEGER N, I
N=0
DO I = 0, 100, 1
N=N+I
END DO
PRINT *, N
END
```

Programming Languages





Cobol

Fortran is an imperative programming language designed for business computing by a committe of computer scientist from academia and industry in the 1960s. It was inspired by earlier languages designed by Grace Hopper.

```
identification division.
program-id. Gauss.
data division.
working-storage section.
01 n pic 9999 value zeros.
01 i pic 9999 value zeros.
procedure division.
perform varying i from 0 by 1 until i > 100
    add i to n
end-perform
display n.
stop run.
```

Pascal

Pascal is an imperative programming language designed for structured programming by Niklaus Wirth in the 1970s. It was widely used for teaching computing in the 1980s.

```
program gauss (output);
var
n : integer;
i : integer;
begin
n:=0;
for i:= 0 to 100 do
   n:= n + i;
   write(n);
end.
```

Ada

Ada is an imperative and object oriented programming language designed for object oriented and structured programming by CII Honeywell Bull in the 1970s. It was named after Ada Lovelace.

```
with Text_Io;
procedure Sum100 is
    N : Natural := 0;
begin
    for I in 1 .. 100 loop
        N := N + I;
    end loop;
    Text_Io.Put_line(Natural'Image(N));
end Sum100;
```

Perl

Perl is a scripting language designed for shell programming by Unisys in the 1980s.

```perl
my $n = 0;
for (my $i = 1; $i <= 100; $i++) { $n = $n + $i; }
print $n;
```

SmallTalk

SmallTalk is an object oriented programming language designed for education by Xerox PARC in the 1970s.

```
m := 0.
0 to: 100 do: [:i | m := m + i. ]
m printNl.
```

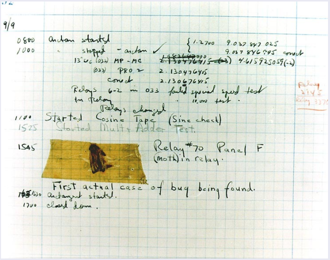
Prolog

Prolog is logic programming language designed for computational linguistic by Alain Colmerauer and Philippe Roussel in the 1970s. It was widely used for artificial intelligence in the 1980s and 1990s.

```
:- sum(100, R), writeln(R)
sum(0, 0).
sum(I, J) :- II is I - 1, sum(II, JJ), J is I + JJ.
```

Computers Err

- Compile-time error: Some errors are caught by the compiler.
- Runtime error: Some errors are caught by the operating system or interpreter or cause the application or system to crash at runtime.
- Some errors simply result in unwanted (and sometimes undetected) behaviours.



Harvard University Mark II Computer group's 1947 log book, entry attributed to
Grace Hopper (December 9, 1906 - January 1, 1992).

Where could the Bug Be?

Errors in hardware, operating systems, compilers, interpreters and commercial application software occur. They are well publicized when discovered and fixed in the following versions (in particular when they can be security threats: see "Top 25 Most Dangerous Software Errors" `http://cwe.mitre.org/top25`).

Hardware errors are the rarest but occur: The Pentium FDIV bug discovered by Professor Thomas R. Nicely in October 1994 - "An error in a lookup table created the infamous bug in Intel's latest processor", by Tom R. Halfhill, BYTE (March 1995).

Errors can be due to interactions between components, for instance in operating systems, the interaction between drivers and applications.

Attribution