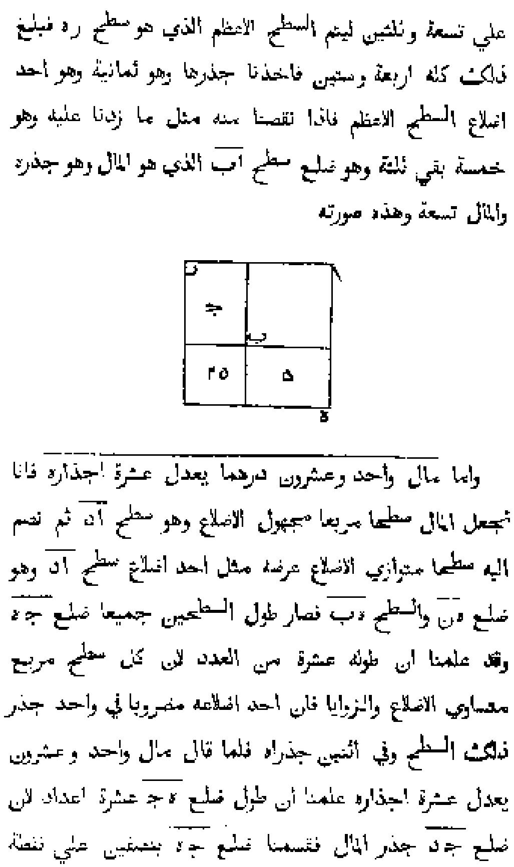
UCT Department of Computer Science
Computer Science 1015F

Creating Software

Aslam Safla <aslam@cs.uct.ac.za>

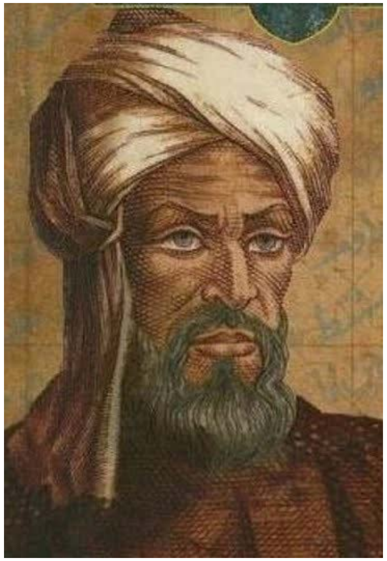
(thanks to Hussein Suleman <hussein@cs.uct.ac.za>)

Muḥammad ibn Mūsā al-Khwārizmī (780-850)



علي تسعة وثلثين ليتم السطح الاعظم الذي هو سطح ره فبلغ
ذلك كله اربعة وستين فاخذنا جذرها وهو ثمانية وهو احد
اضلاع السطح الاعظم فاذا نقصنا منه مثل ما زدنا عليه وهو
خمسة بقي ثلثة وهو ضلع سطح اب الذي هو المال وهو جذره
والمال تسعة وهذه صورته

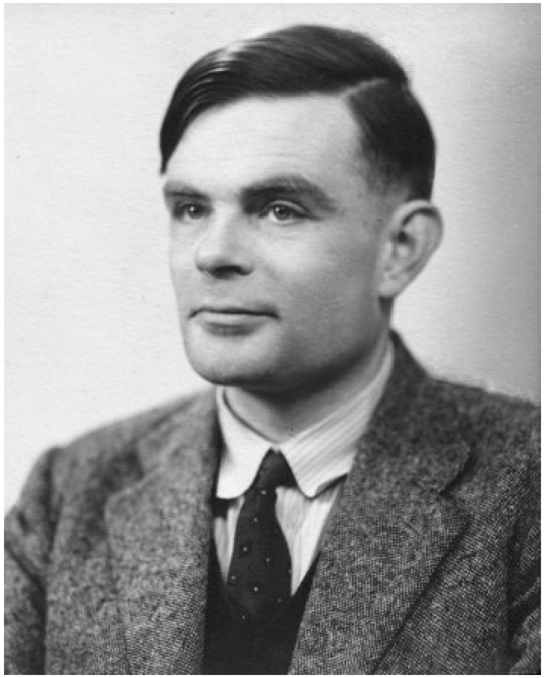
واما مال واحد وعشرون درهما يعدل عشرة اجذاره فانا
نجعل المال سطحا مربعا مجهول الاضلاع وهو سطح اد ثم نضم
اليه سطحا متوازي الاضلاع عرضه مثل احد اضلاع سطح اد وهو
ضلع هن والسطح هب فصار طول السطحين جميعا ضلع جه
وقد علمنا ان طوله عشرة من العدد لان كل سطح مربع
متساوي الاضلاع والزوايا فان احد اضلاعه مضروبا في واحد جذر
ذلك السطح وفي اثنين جذراه فلما قال مال واحد وعشرون
يعدل عشرة اجذاره علمنا ان طول ضلع هج عشرة اعداد لان
ضلع جد جذر المال فقسمنا ضلع جه بنصفين علي نقطة



- □Persian mathematician
- □Early work on Algebra
- ■al-Kitāb al-mukhtaṣar fī ḥisāb al-jabr wal-muqābala
- □Later work on Algorithms
- ■Translated as: Algoritmi de numero Indorum

Alan Turing (1912-1954)

- Father of Computer Science
- Major contributions:
  - Defined what can be called an algorithm and showed the universality of algorithms.
    - Church-Turing thesis
  - Defined a Universal Computer
    - Turing Machine
  - Defined how we test for AI
    - Turing Test
  - Part of WWII code-breaking team that deciphered Enigma messages.
- Harassed by UK govt because he was gay
  - committed suicide in 1954.



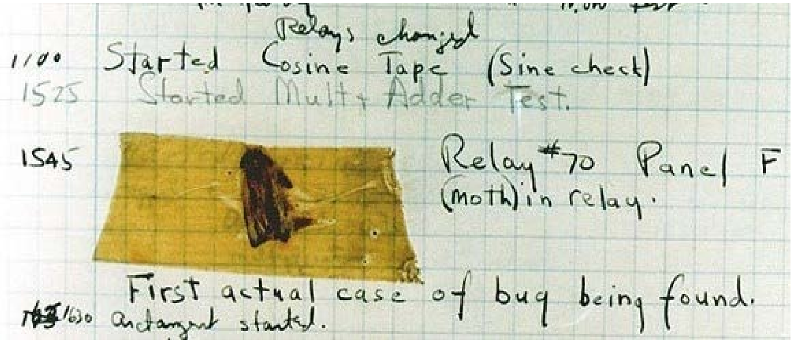
Some Turing Award Winners

- Highest honour in computer science

- 1981: Codd (relational databases)
- 1983: Thompson and Ritchie (UNIX)
- 1997: Engelbart (mouse!)
- 1999: Brooks (software engineering)
- 2002: Rivest, Shamir and Adleman (RSA encryption)
- 2003: Alan Kay (Smalltalk - OOP language)
- 2004: Cerf and Kahn (Internet)
- 2008: Barbara Liskov (programming language principles)
- 2015: Diffie and Hellman (cryptography)
- 2016: Sir Tim Berners-Lee (WWW)

Grace Hopper (1906 - 1992)

- Mathematician, computer scientist, Admiral in US Navy
- Major contributions:
  - High-level programming language
    - COBOL
  - Invented first compiler
  - Conference and awards in her honour
  - Popularized the term “bug”
  - All this and she started coding at age 37...



Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
1630 Antangent started.

Women in Computer Science

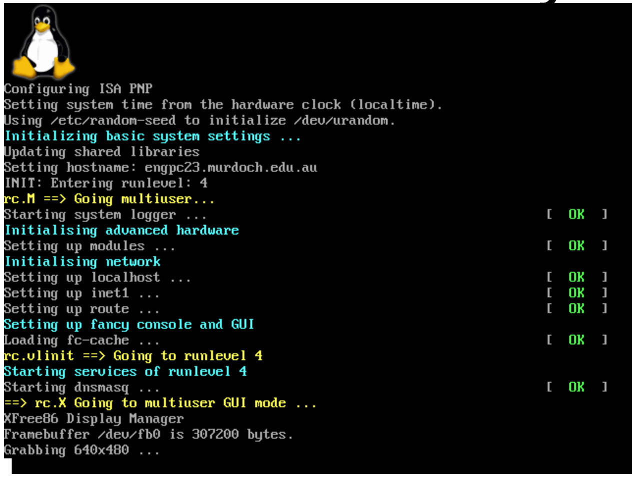
- Ada Lovelace (1815 - 1852)
  - First computer program on Babbage's analytical engine
- Margaret Hamilton (1936 -)
  - Lead s/w designer for Apollo missions
- Adele Goldberg (1945 -)
  - Part of the Smalltalk team at PARC
- Anita Borg (1949 - 2003)
  - Advocacy for technical women

What does the CPU understand?

- **Machine Code** is the only language a CPU understands directly!
- Each instruction is a sequence of numbers.
- On x86 CPUs, instructions have variable lengths.
  - some are 2 numbers, some 3 numbers, etc.
- For example:
  - 180 76 = store value of 76 in special CPU variable AH
  - 205 33 = call OS function (if AH=76, this means quit)

The Operating System

- Manages resources on computer.
- Executes on startup (boot):
  - BIOS ROM has instructions to load OS - fixed in hardware!
  - Disks are checked in order defined by hardware.
  - If OS machine code is on a disk, load it into memory and start execution.
  - OS takes over and allows users to select and run their programs until computer is shut down.



```
Configuring ISA PNP
Setting system time from the hardware clock (localtime).
Using /etc/random-seed to initialize /dev/urandom.
Initializing basic system settings ...
Updating shared libraries
Setting hostname: engpc23.murdoch.edu.au
INIT: Entering runlevel: 4
rc.M ==> Going multiuser...
Starting system logger ...                                          [  OK  ]
Initialising advanced hardware
Setting up modules ...                                              [  OK  ]
Initialising network
Setting up localhost ...                                            [  OK  ]
Setting up inet1 ...                                                [  OK  ]
Setting up route ...                                                [  OK  ]
Setting up fancy console and GUI
Loading fc-cache ...                                                [  OK  ]
rc.vlinit ==> Going to runlevel 4
Starting services of runlevel 4
Starting dnsmasq ...                                                [  OK  ]
==> rc.X Going to multiuser GUI mode ...
XFree86 Display Manager
Framebuffer /dev/fb0 is 307200 bytes.
Grabbing 640x480 ...
```

Low Level Languages

- Machine Code is a low level language.
  - ONLY language CPU can understand.
  - Different MC for every CPU!
- Low level languages are easier for a machine to understand, and often difficult for a human.
- Assembly language expresses machine code symbolically - so humans can write programs more easily.
- Example (quit a program):
  - decimal: 108 76 205 33
  - hexadecimal: B4 4C CD 21
  - assembler:

```
MOV AH,4Ch
INT 21h
```

Assembler Programming

```
Memory     machine        assembly                comments
location   language       language
            (HEX)
21AA:0100 B409           MOV     AH,09           ;display string of characters
21AA:0102 BA1701         MOV     DX,0117         ;point to string
21AA:0105 CD21           INT     21              ;do it
21AA:0107 B401           MOV     AH,01           ;keyboard input function
21AA:0109 CD21           INT     21              ;do it
21AA:010B B44C           MOV     AH,4C           ;exit function
21AA:010D 2C30           SUB     AL,30           ;convert to number
21AA:010F 7EF6           JLE     0107            ;jump to 107 if < "1"
21AA:0111 3C09           CMP     AL,09           ;compare to "9"
21AA:0113 7FF2           JG      0107            ;jump to 107 if greater
21AA:0115 CD21           INT     21              ;do exit
```

Source: h*ttp://www.kyphilom.com/www/txt/compdo.txt*

Programming in Assembler

- Write program in text editor.
- Save program in file.
- **Assemble** source code into object/machine code.
  ```
  tasm hello.asm
  ```
- Optionally link multiple files together and create executable understood by OS.
  ```
  tlink hello.obj
  ```
- Execute application in OS.
  ```
  hello.exe
  ```

Pros/Cons of Assembler

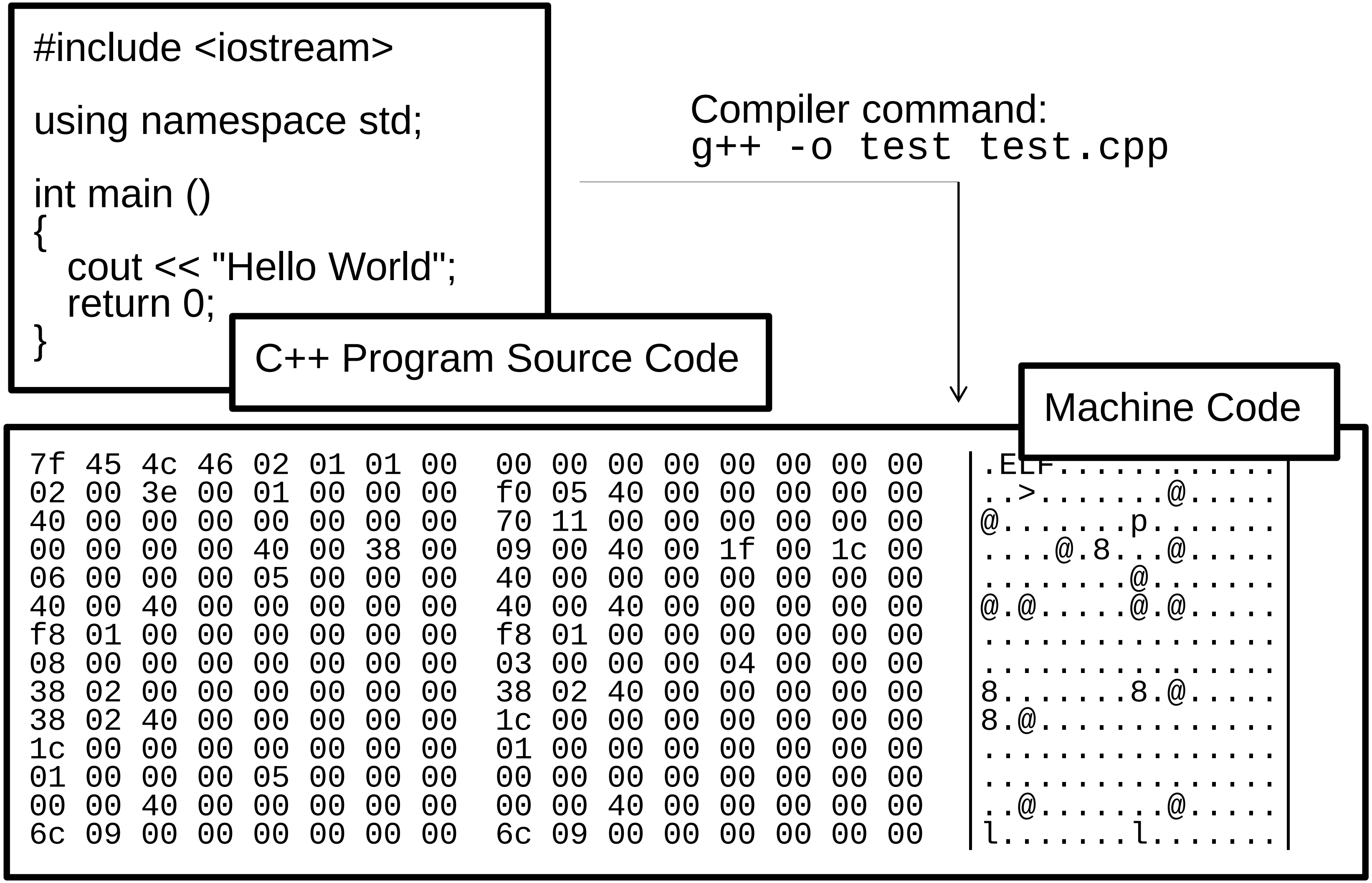
- Pros
  - Matched machine code so can do whatever CPU can do.
  - Very fast execution of programs.
  - Can be used on obscure CPUs.
- Cons
  - Difficult to program.
  - Programs are very long.
  - Programming is slow process and prone to errors.

High Level Languages

- **High level languages** are easier for humans to understand.
- We need to convert programs in high level languages to low level languages so computers can understand.

- 2 common approaches:
  - Compile
  - Interpret

Compilers (C++)



```
#include <iostream>

using namespace std;

int main ()
{
   cout << "Hello World";
   return 0;
}
```

C++ Program Source Code

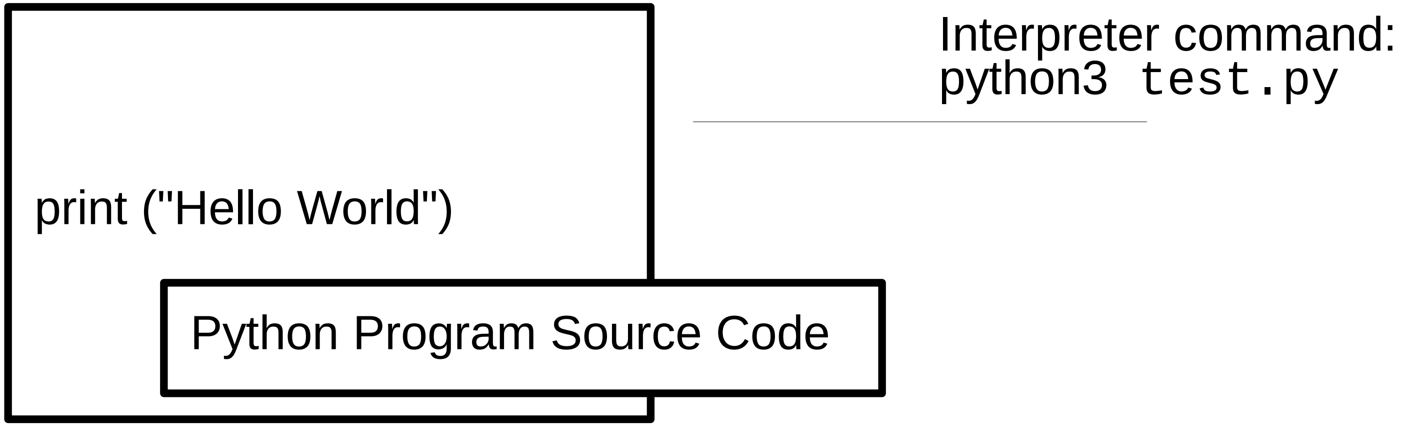
Compiler command:
`g++ -o test test.cpp`

Machine Code

```
7f 45 4c 46 02 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
02 00 3e 00 01 00 00 00  f0 05 40 00 00 00 00 00  |..>.......@.....|
40 00 00 00 00 00 00 00  70 11 00 00 00 00 00 00  |@.......p.......|
00 00 00 00 40 00 38 00  09 00 40 00 1f 00 1c 00  |....@.8...@.....|
06 00 00 00 05 00 00 00  40 00 00 00 00 00 00 00  |........@.......|
40 00 40 00 00 00 00 00  40 00 40 00 00 00 00 00  |@.@.....@.@.....|
f8 01 00 00 00 00 00 00  f8 01 00 00 00 00 00 00  |................|
08 00 00 00 00 00 00 00  03 00 00 00 04 00 00 00  |................|
38 02 00 00 00 00 00 00  38 02 40 00 00 00 00 00  |8.......8.@.....|
38 02 40 00 00 00 00 00  1c 00 00 00 00 00 00 00  |8.@.............|
1c 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  |................|
01 00 00 00 05 00 00 00  00 00 00 00 00 00 00 00  |................|
00 00 40 00 00 00 00 00  00 00 40 00 00 00 00 00  |..@.......@.....|
6c 09 00 00 00 00 00 00  6c 09 00 00 00 00 00 00  |l.......l.......|
```

Interpreters (Python)

- Interpreter reads each statement and executes an equivalent set of machine code instructions.
- Compared to compiled programs:
  - Easier to program - no compile step.
  - Programs run slower - source code must be processed every time.



```
print ("Hello World")
```

Python Program Source Code

Interpreter command:
`python3 test.py`

Types of High Level Languages

- Procedural/Imperative, Object-oriented - programs are specified as exact sets of instructions to execute.
  - Python, Java, C++
- Declarative/Logic - programs are rules and facts that are processed by an engine.
  - Prolog, XSLT
- Functional - programs are collections of functions that are applied and composed to solve problems.
  - Haskell, Lisp