- **LECTURE 2**
  - History of computing: What led to Unix, C, and PC/
  - Linux OS
  - About Internet and Networking
  - Languages of High Performance Computing
  - Physics of integrated circuits and the transformation of civilization in the last 50 years

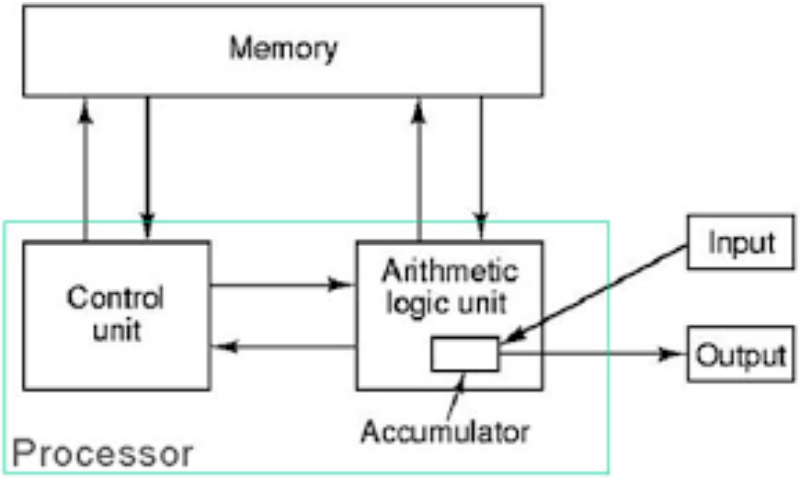
in next lectures:

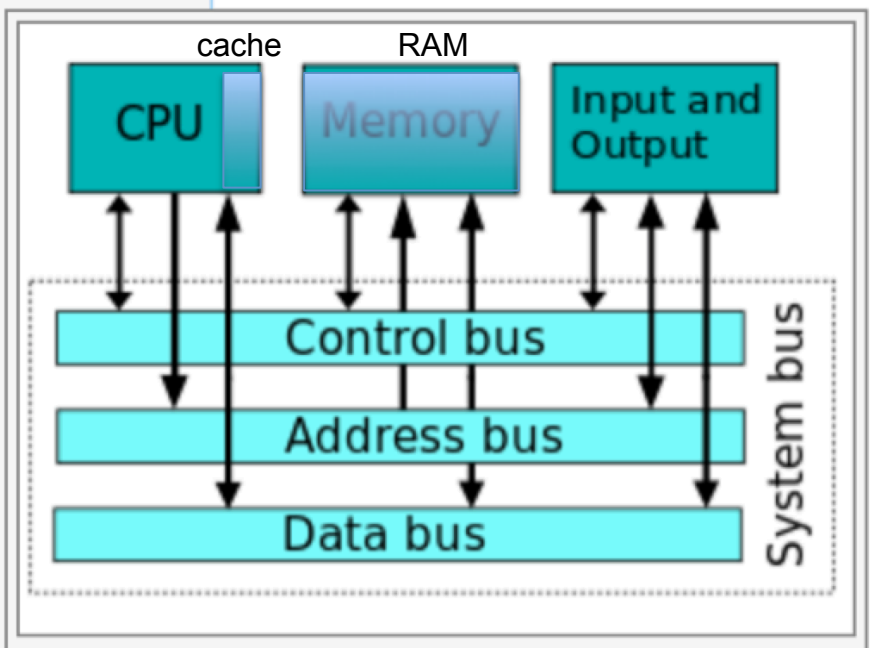
  - Supercomputing today (incl. at UTSC)

Literature

1. Top 500 supercomputer site https://www.top500.org/
2. Norbert Schoerghofer "Lessons in Scientific Computing" (2019)
3. Joshua Izaac & Jingbo Wang, "Computational Quant. Mech." (Undergrad. Lecture Notes in Physics, Springer, 2018) – chapters on Python and Fortran
4. Whole page of suggestions for you http://planets.utsc.utoronto.ca/~pawel/PHYD57/index-lit-x.html

Von Neumann computer implements the logic of *Turing machine* – a model, where data and program are stored in the same memory,





modern computer architecture:

Memory = Random Access Memory (volatile, transistors need some power to keep states 0 & 1)

Storage = Nonvolatile Memory
(hard disk, solid state mem., earlier: magnetic tape, punched paper) work like I/O

[Several buses on motherboards, one of them is PCIe = data bus for peripheral devices + peripheral devices (GPU, MIC, NIC, storage, etc.)]

1950s to 1960s

At the end of 1950s and beginning of 1960s many advances gradually appeared:

- magnetic tape as mass storage of von Neumann architecture:
- separate operating and mass storage, processor, data and program in the same memory
- magnetic drums playing the role of harddisks, developed later by IBM
- binary digits won with decimal digits
- from among different data formats, 8-bit bytes started to emerge as winners
- they simply were handy for character (text) processing
- instead of setting up the computer by hand or from punched paper media, the programming was done from magnetic media, and the code was no longer the low-level machine code (fairly elementary operation commands understood by the processor.
- instead, *compilers* of *high-level programming languages* appeared

- *Compiler* = program that checks and translated your program into the set of low-level instructions for processor (still readable by humans, but program in such assembly language was ~20 times longer than the high-level code in language similar to English+math symbols). Finally the so-called linker (part of compiler in modern times) was translating the assembly language into a binary executable, containing a stream of 00011100110101001000101010 11... , which include everything needed for computation: part of data, instruction codes and memory addresses.

```
# mark_description "Intel(R) Fortran Intel(R) 64 Compiler XE for applications running
on Intel(R) 64, Version 15.0.3.187 Build 2";
        .file "fortran+om-laplace-sp.f90"
(...)
 movsd     %xmm1, data_mp_t1_(%rip)                              #113.5
        movl      1950048+data_mp_grid_(%rip), %r12d             #114.2
        jne       ..B1.18       # Prob 50%                       #115.10
                                # LOE rbx r14 r15 r12d r13d xmm1
..B1.13:                        # Preds ..B1.12
        movl      $-1, %esi                                      #115.14
        lea       32(%rsp), %rdi                                 #115.14
        movq      $0x1208384ff00, %rdx                           #115.14
        movl      $__STRLITPACK_9.0.3, %ecx                      #115.14
        xorl      %eax, %eax                                     #115.14
        lea       232(%rsp), %r8                                 #115.14
        movq      $0, (%rdi)                                     #115.14
        movq      $15, 232(%rsp)                                 #115.14
        movq      $__STRLITPACK_8, 240(%rsp)                     #115.14
        movsd     %xmm1, 288(%rsp)                               #115.14
call      for_write_seq_lis                                      #115.14
                                # LOE rbx r14 r15 r12d r13d
        movsd     288(%rsp), %xmm1
#pxor      %xmm0, %xmm0                                          #115.14
        cvtsd2ss  %xmm1, %xmm0                                   #115.14
```

Assembly code – processor's language

Too low-level; we will not code in assembler!

Computing in 1950s and 1960s

- Hardware and software, previously inextricably bound, now became independent when high level programming languages appeared that are closer to human mathematical language than assembly language (which processors understand, when translated into binary form)

- The first widely used computer language which was hardware platform-independent was **FORTRAN** (1957) = Formula Translator. It was created at IBM by a small team led by John Backus, for computer called IBM 704.

- This language was loved by academia and the first computer science departments (computer science just trying to emerge and define itself in opposition to EE). Fortran resembled *meta-code* and English language. It produced executable programs running as fast as hand-coded *assembly code.*

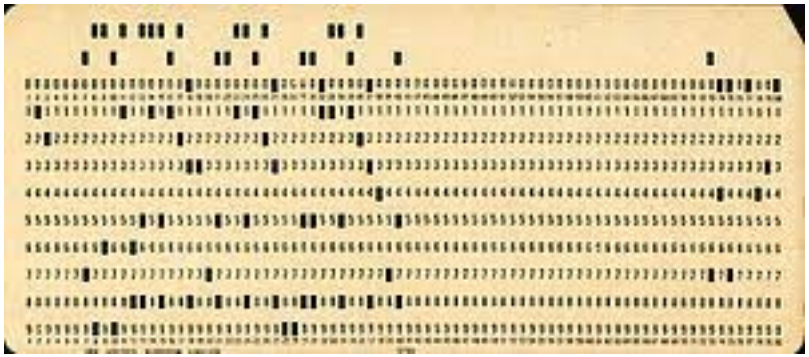
- Then **COBOL** was mandated by the U.S. government on all machines it was purchasing. (COBOL = Common Business Oriented Language.) COBOL is now pretty much extinct, while

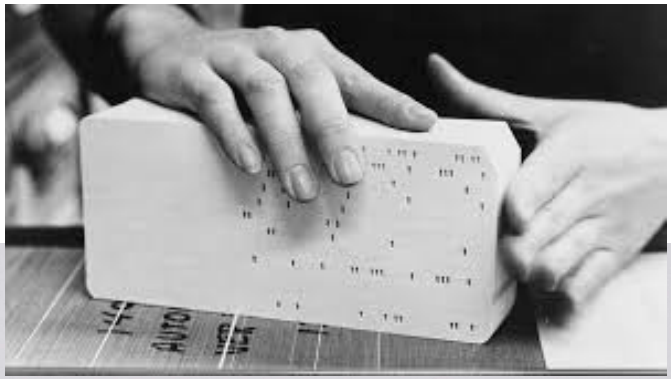
- Fortran (77 ver. & new Fortran 90, 95, 2003, 2018 ver.) thrives in physical science and engineering, at the supercomputer centers, but is little known elsewhere. In business and at CS departments, Java, Python prefer 10s of other languages.

- Crucial parts of *Matplotlib* and *Numpy* of Python were first written in Fortran (& C).

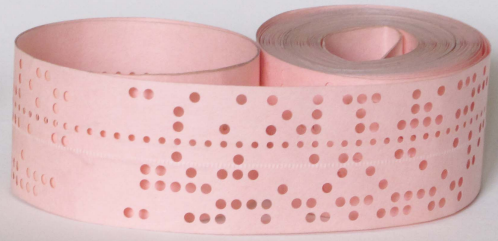
Mid-1960s and 1970s

IBM (International Business Machines Co.) grew when it transferred from decks of punched cards (very popular among businesses)





... and perforated tape storing programs



to magnetic media: tape, drums, disks

Q: How many times more data is stored on a 2013 hard-disk than on 1956 version?

Mid-1960s to 1970s

IBM captured 70% of global computer sales, mostly with very expensive ***mainframes*** (or supercomputers, bought by big firms, government centers and the universities). The rest of the market was served by Univac, Computer Data Corp., Sci. Data Systems, and Cray, which tried to emulate IBMs scale, salesforce and success.

DEC = Digital Equipment Corporation appeared as a small company in 1960s, located in old Abbotts Mills near Boston, benefitting from MIT connections. DEC, later called **Digital**, started a quiet revolution in the market by offering:

- much smaller ***minicomputers***, costing much much less than mainframes. For instance, model PDP-8 costed $18k instead of IBM/360 for $2 million.
- Similar speed on some tasks, while much less data storage capability etc.
- Circuits in 1960s were non-integrated (cf. below) but fast-switching transistor-based, which allowed fairly large computational speeds

- Innovation in comp. architecture: *the interrupts* in CPU operation to serve input/output requests from the peripheral devices (printers, storage, keyboards, monitors)

- Different & much less pretentious culture in workplace and in sales

- OEMs (Orig. Equip. Manufacturers) appeared when Digital opened up their architecture, encouraged modifications

1970s and later...

- PDP-7 on which Unix was born, see below, 4k 18-bit op. memory
- **PDP-8** (left picture) still little RAM but fast and cheap

- **PDP-11** (right fig.)
- 128k RAM
- 2 MB removable hard disk
- > 30 kFLOPS arithmetic speed
- interactive consoles, time-sharing operating system, as opposed to batch input on mainframes

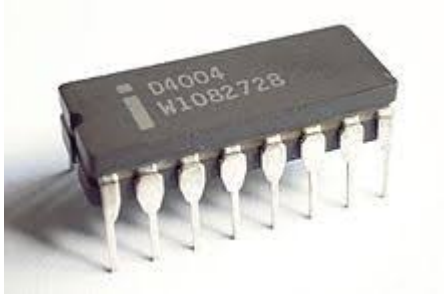
- Languages such as ALGOL, Pascal, and other became popular but later faded away. IBM's PL/I (Programming Lang. I) was a commercial failure.
- **DEC** offered the first **UNIX operating systems** on PDP-line computers
- **C** appeared in 1973 and later **C++**, high-level languages that unlike numerical computation-oriented Fortran for "number crunching", were not meant for scientific simulations. They were great for writing operating systems (fully C-based Unix in 1973, later **Linux** in 1990s, which is a modified Unix), and for interfacing at low level with hardware (low-level here means using elementary, simple instructions).

THE C PROGRAMMING LANGUAGE

Interactive use of Digital's PDP-11 minicomputers (here shown in year 1977) was a harbinger of and a model for personal computers invented right then and getting huge popular in 1980s.

Apple Computer Inc. was founded in 1976 by Steve Jobs and Steve Wozniak, with a vision to improve on that achievement & go much further toward

- miniaturization allowed by small ***microprocessors***, and
- personalization: personal computing, personal publishing and all other things personal & social on Apple, Mac, ibook, imac, macbook, itunes, iphone, ipad, ...



- **Programmable calculator and PC revolution** based on *semiconductor technology in microchips* or *microprocessors.*
- In 1971-1974, **Intel** (4004; 8008, 8-bit) and **Motorola** (8-bit 6800)
- Intel 8086 was a 16-bit CPU (central processing unit) from 1978,
  → "x86_16" (today: x86_64, i.e. 64-bit processors).
- Motorola 68000 was a 16/32-bit CPU from 1979
- In 1980s and 1990s computers spread outside academia, big business and military. Microcomputers or desktop Personal Computers (PCs) moved into every house and school, into the backpacks as laptops, and finally into our pockets as smartphones, not for scientific computation but rather for a simpler task(?) of communication (networking). This interesting history is outside the scope of our course, even though for your computing will likely choose a laptop

- Simple languages like BASIC, and in late 1970s/early 1980s the scripting languages MATLAB, and IDL have appeared. One of them is currently hugely popular:
- **Python** *interpreter* was created in 1989 and became popular in 1990s and 2000s.
- Creator, Guido van Rossum, was named the Benevolent Dictator for Life. Now the project is guided by Steering Council. Current version: Python3, appeared in 2008.

- In the 2004 a new and still not widely known but powerful and elegant language Julia was created by professors and students at MIT. Produces multithreaded code often executing almost as fast as C/Fortran. One day it may replace Python in science applications, though not necessarily in other areas such as business. Computer Sci. & Computational Sci. have *different* goals & fashions, too...

1980s and 1990s From Unix to Operating Systems as community projects

AT&T Bell Labs in 1969-1971: Thomson and Ritchie use PDP-7 to create Unix OS.

In 1973 rewritten in new language C, and unveiled.

At UC Berkeley in late 70s a strain of Unix called BSD appears (Berkeley Software Distribution)

Commercial Unix:
Sun Micro:Solaris
Hewlett-P:HP-UX
SCO: Xenix
AT&T: System V
SGI: Irix

GNU Project starts in 1984, with a goal of a free version of Unix. Created many programs but failed to produce an OS ***kernel***

In 1991 in Finland, Linus Torvalds begins creation out of GNU, BSD and X-windows software of a Unix-like OS ***kernel.*** Calls it ***Linux***

1994-2007 **Linux** OS & flavors/distributions = packaged OS filled with slightly different components appear:
Red Hat, Fedora, FreeBSD, Caldera, Debian, SuSE, Mint, Gentoo, Ubuntu, CentOS, etc.
Also: MacOS, and Linux-kernel OS's with quite different GUI = graphical user interface: Android (2007), Chrome (2009)

We will return to the further historical development of hardware a bit later, after we explore the operating system types that conquered the world (not only of HPC, but also of web servers). Incidentally, learn some Internet history here.

Sobell – A practical guide to Linux Commands – 2018
Gedris – Intro to Linux Command Shell for Beginners-2003

Good web resources:
www.linfo.org - on various topics about Linux, see all the links on that page, among others
www.linfo.org/command_index.html - list commands with further explanations on the use
www.linfo.org/how-to_index.html - how to guides
Also, see the Links section on our course web page.

Well.. beside its history, how does the networking we practice today actually work?

Networking is a separate magic/technology.

Find out yourself by... using it to see the videos below!

Series of informative videos about networking

2. https://www.youtube.com/watch?v=kn7ei2ENJbI
3. https://www.youtube.com/watch?v=fH-fiVm5__o
4. https://www.youtube.com/watch?v=j3LXv7RNFLY
5. https://www.youtube.com/watch?v=mGRClHHgNdk
6. https://www.youtube.com/watch?v=t1MvzaScl9k
7. https://www.youtube.com/watch?v=fyS3QdYuMZI
8. https://www.youtube.com/watch?v=uTE48tWIxuI
9. https://www.youtube.com/watch?v=Cx7foWGm5fo
10. https://www.youtube.com/watch?v=o4xkvyZnhj4
11. https://www.youtube.com/watch?v=O48yc2nYBRA
12. https://www.youtube.com/watch?v=_Cv5OnPHo-k
13. https://www.youtube.com/watch?v=b_9Dg0QYJUg