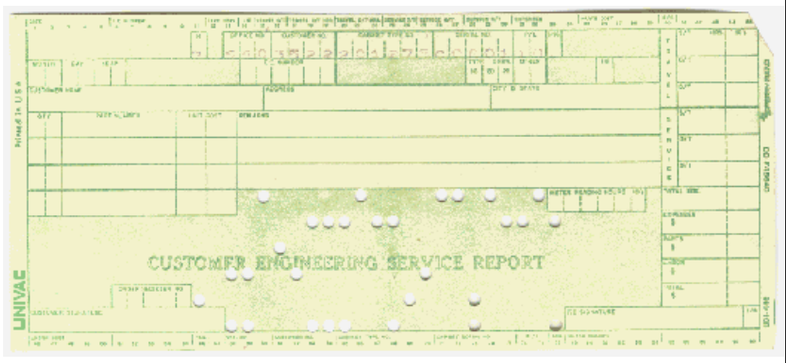
Operating Systems I - Intro, History

Beuth Hochschule
Summer Term 2014

The First Computer(s)

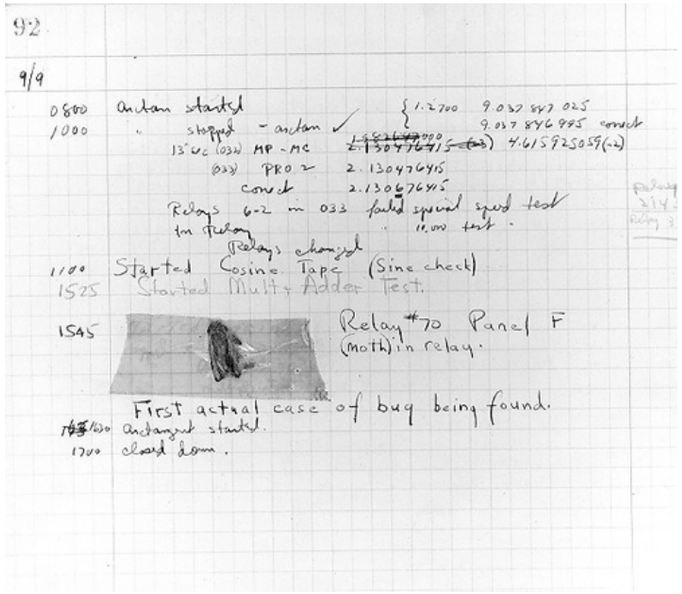
- 1801: Power loom driven by wooden punch cards
- 1822: Steam-driven **analytical engine** by Charles Babbage
  - Mechanical decimal stored-program computer, programmable by punch cards, support for calculation and conditional statements
  - Remained unbuilt; Ada Byron invented anyway subroutine and looping as programming concepts
- 1890: U.S. census supported by **Hollerith desk** - punch card reader, counting units, wall of dial indicators
  - Built by **Tabulating Machine Company**, which eventually became **International Business Machines**
  - Invented the idea of output punch cards independent from Babbage



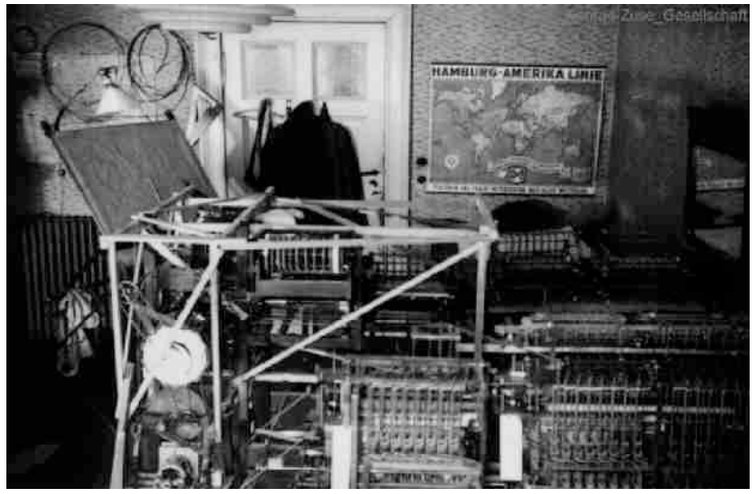
(C) computersciencelab.com

The First Computer(s)

- 1944: **Harvard Mark I** developed in partnership between Harvard and IBM (ballistic firing tables)
  - First programmable digital computer made in the U.S.
  - Constructed of switches, relays, rotating shafts, clutches
  - **Grace Hopper** found the first computer bug, invented the precessor of COBOL and the first compiler



- 1941: Konrad Zuse completed the work on the **Z3**
  - First programmable electromechanical computer
  - Punch film for program and data (lack of paper)
  - Mapping of Boolean algebra to relays, developed independently from original Shannon work

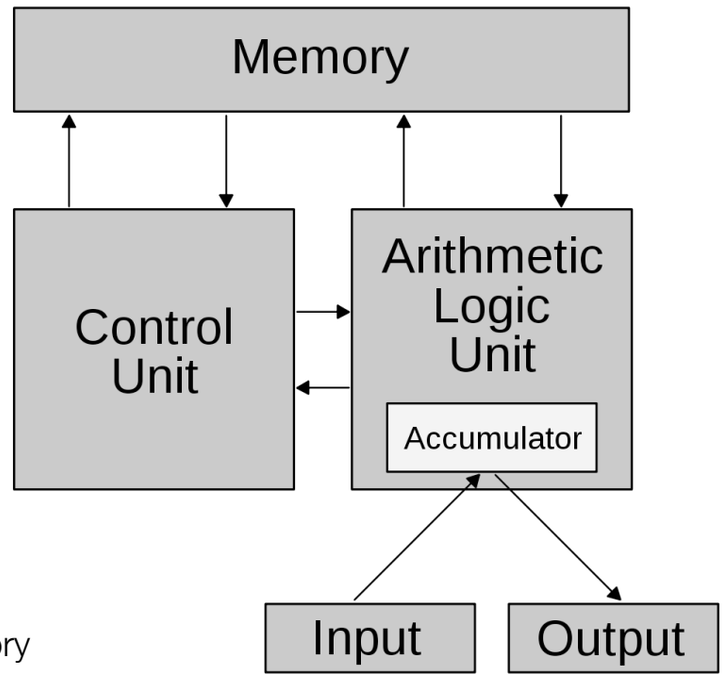


(C) computersciencelab.com

Von Neumann Architecture

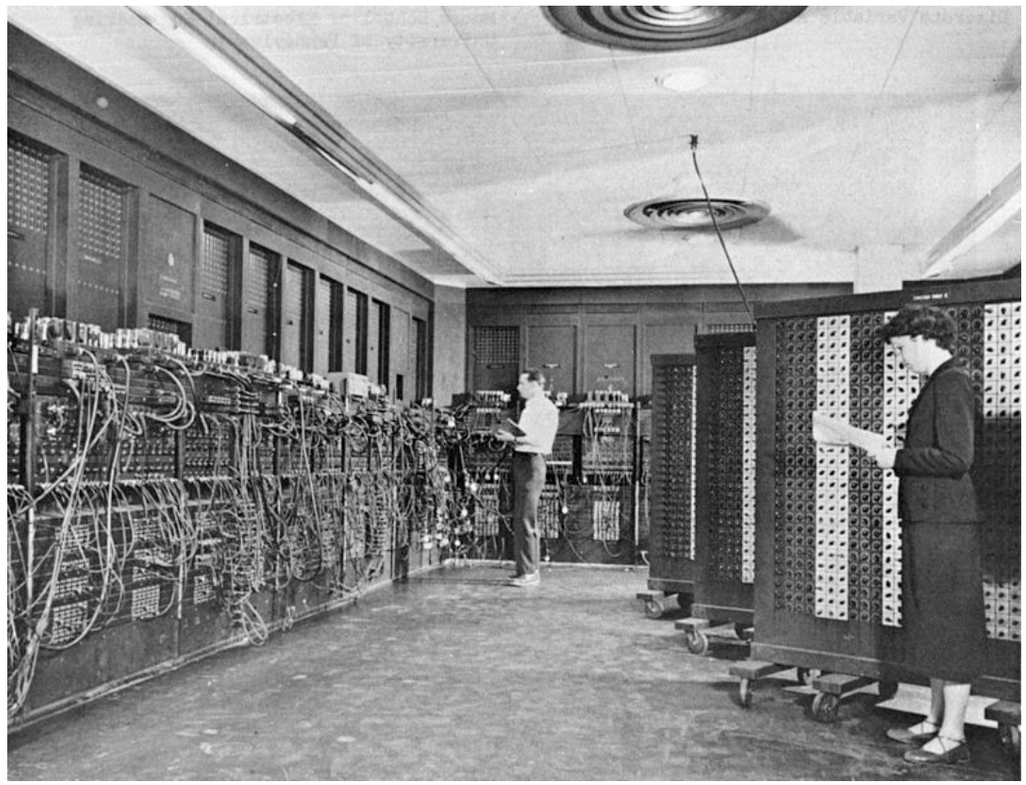
- 1946: **ENIAC** as first fully electronic computer in the U.S.
  - No program memory, re-wiring for each program
- **EDVAC**: Revolutionary extension with a *stored program computer* concept by *John von Neumann*
  - Memory contains both the program and the data
  - Introduction of a general purpose computer concept

(C) Wikipedia



Serial Processing

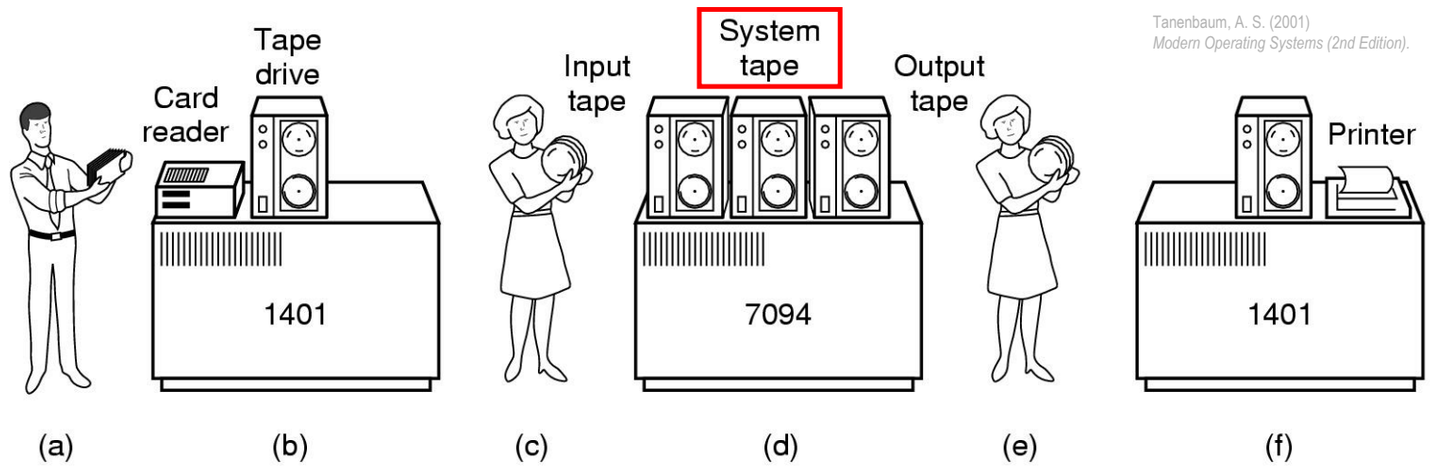
- Computers from 1940 to 1955 were able to perform serial processing
  - Programs for the machine (relays, vacuum tubes) written in assembly language
  - Console with display lights, switches, punch card reader / plugboard, printer
  - Re-wiring or punch card reading necessary for filling the program memory
  - Program had complete control of the machine for the entire run-time
  - Manual reservation, user either finished early (wasted time) or couldn‘t debug their problem
  - Long setup time -
    Job may involve running the compiler program first and feeding in the output again



Batch Processing

- IBM 1401 - October 5th, 1959
  - IBM‘s first affordable general purpose computer, i.e. for accounting
  - 1401 Processing Unit, 1402 Card Read-Punch (250 cards/minute), printer

- First concepts of batch processing for multiple job input cards
  - **Operator** loads **monitor** to run batched jobs from a prepared input tape

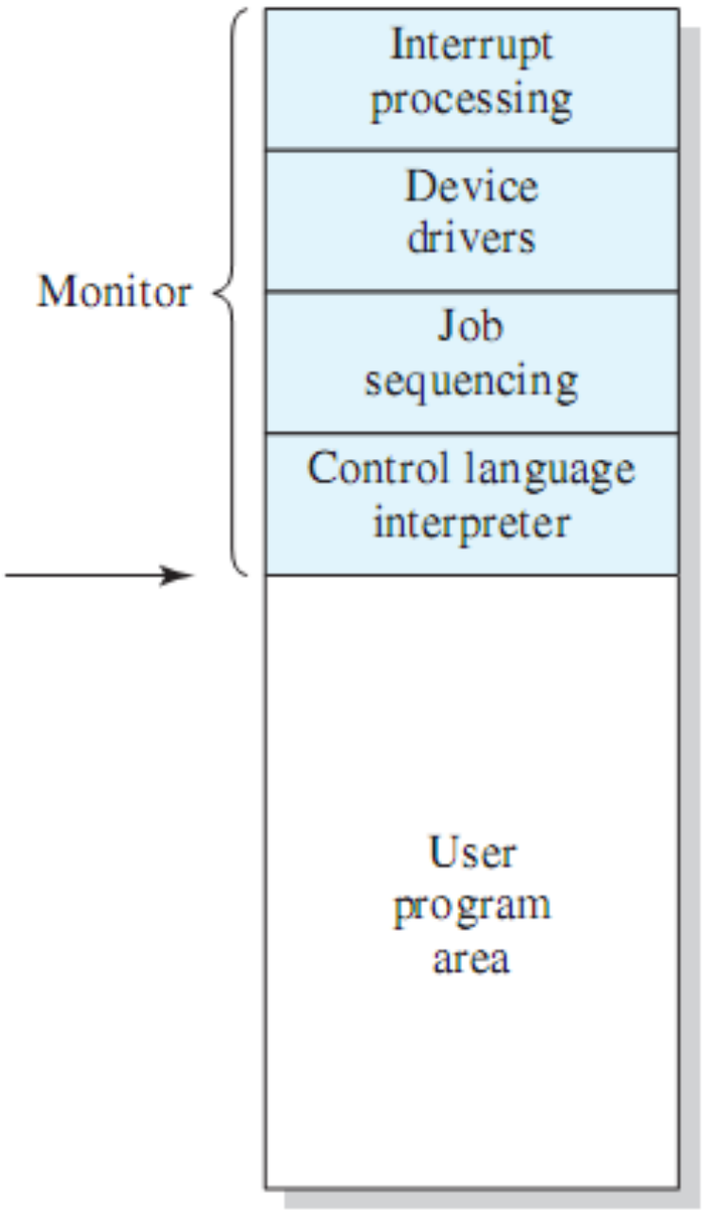


Tanenbaum, A. S. (2001)
Modern Operating Systems (2nd Edition).

- Programs are constructed to branch back to the monitor after termination (early Fortran language)
- First version of a **scheduler**, better utilization of the extremely expensive hardware

Batch Processing

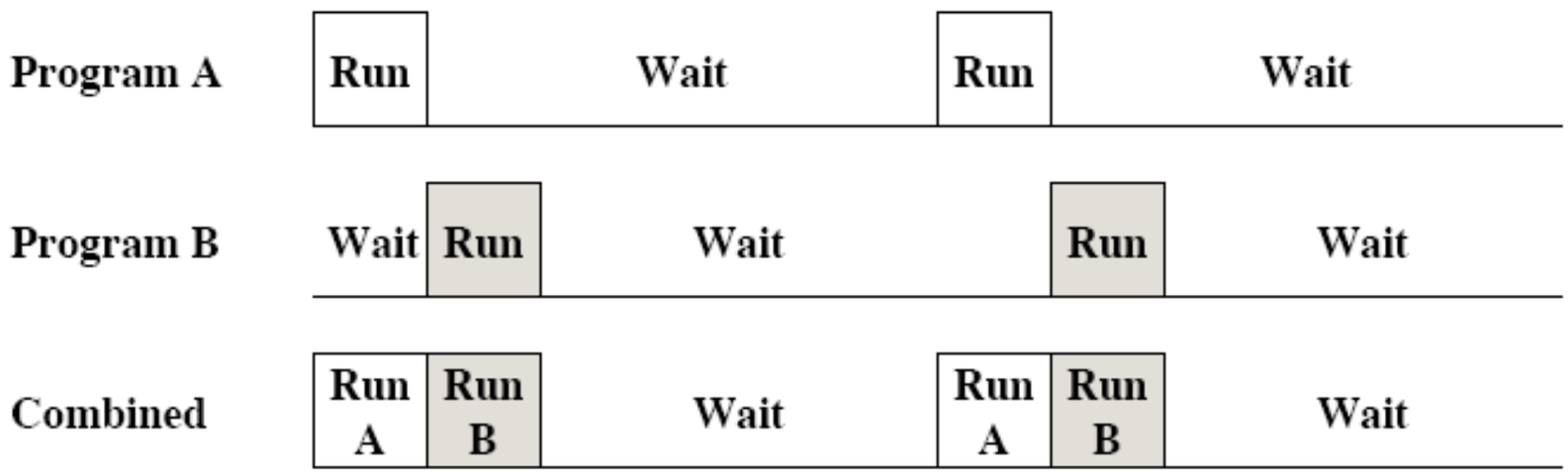
- A **job control language** (JCL) operates the monitor application
  - Instructions about the compiler to use, data to work on etc. (Fortran prefix $)
  - Early version of system calls
- Monitor needed to switch between itself and the application
  - Resident monitor parts always in memory
  - Demands on hardware: **memory protection, timer, privileged instructions for I/O**
- **User mode vs. monitor mode** (‚system mode‘, ‚kernel mode‘, ‚supervisor mode‘)



(C) Stallings

Multi-Programming

- Batch processing is nice, but jobs still spend too much time on I/O completion
- Idea: Load two jobs into memory at the same time
  - While one is waiting for I/O results, switch to the other one
- Multiplexing of resources between a set of jobs became a basic monitor / operating system task -> **multi-programming** or **multi-tasking**
- Goal: Maximize CPU utilization by sacrificing memory



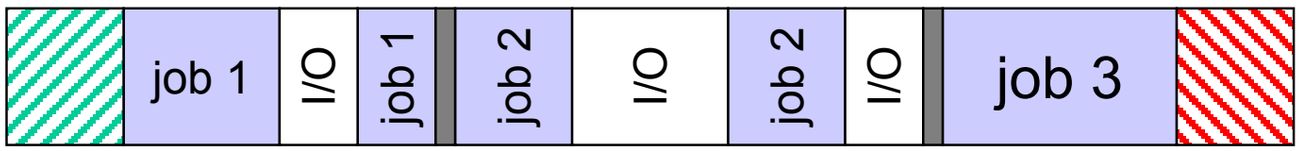
(C) Stallings

Multi-Programming

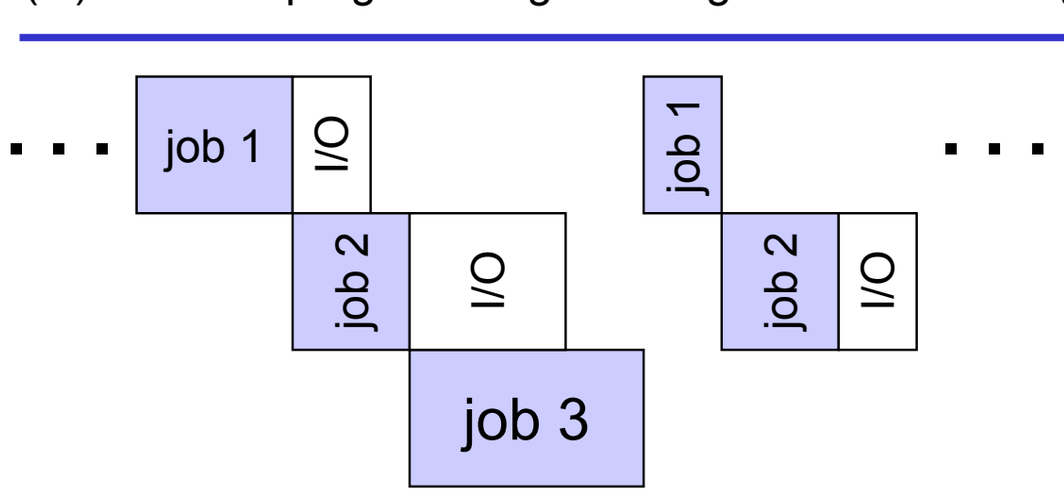
(a) serial uniprogramming

human overhead

(b) batch uniprogramming



(b') batch uniprogramming showing actual CPU usage and I/O wait



(c) multiprogramming

human operator's setup (mount tape, etc.)

human operator's takedown (unmount)

spooling

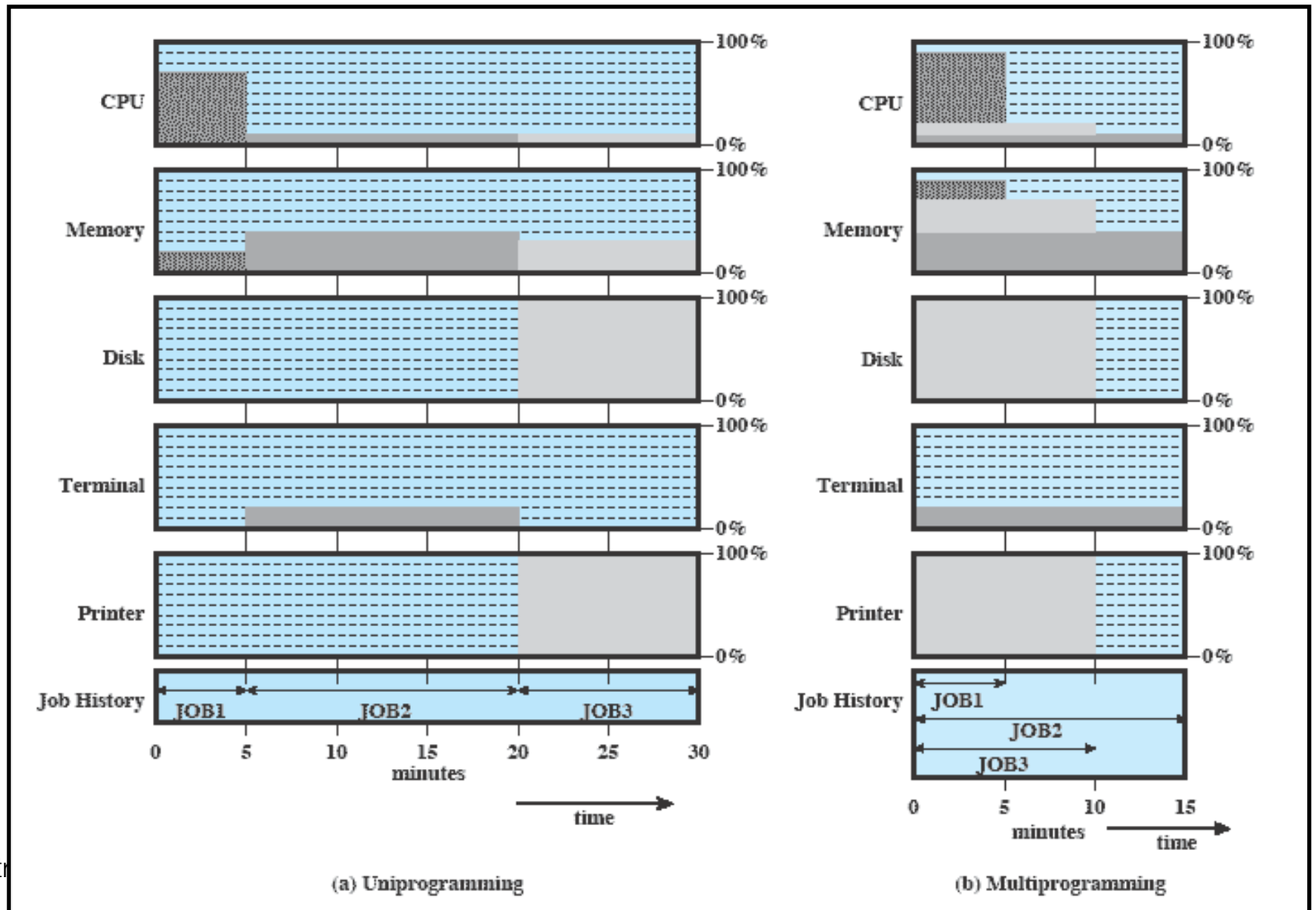
Evolution of CPU utilization

(C) CS446/646

Multi-Programming

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

	Uniprogramming	Multiprogramming
Processor use	20%	40%
Memory use	33%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min

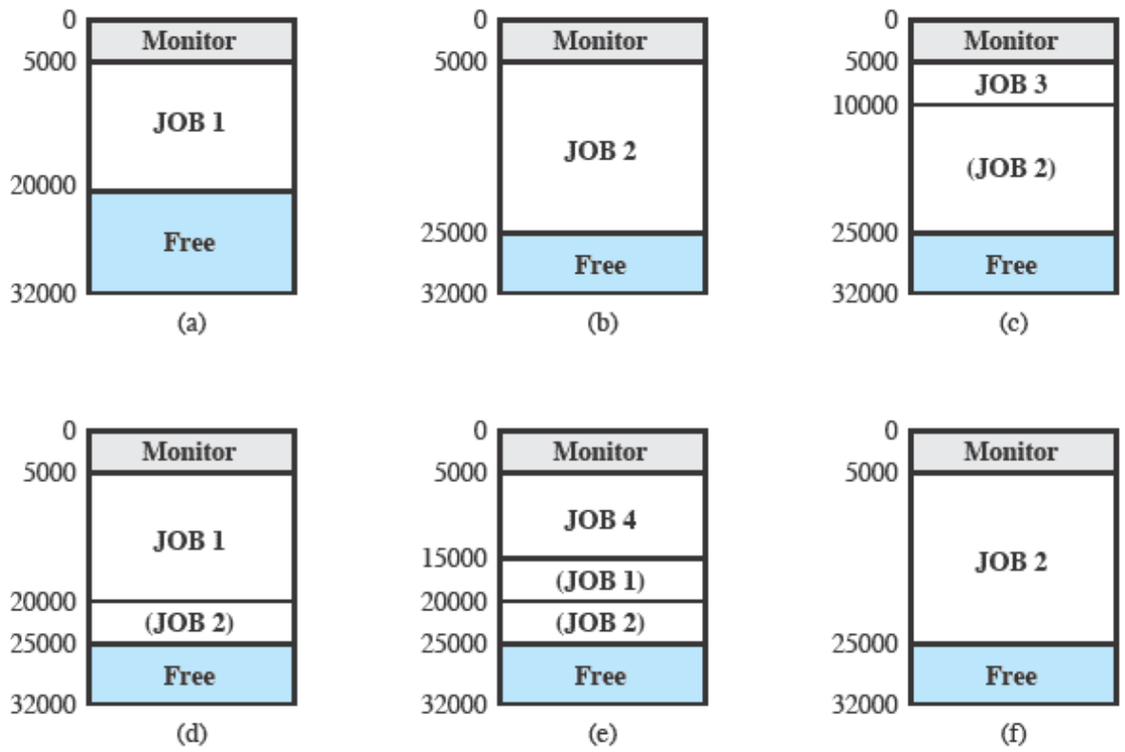


Time Sharing

- Users started to demand interaction with their program, e.g. for retry on errors
  - Perform multi-tasking, but act like the machine is solely used
- Advent of **time-sharing** / **preemptive multi-tasking** systems
  - Goal: Minimize single user response time
  - Extension of multi-programming to multiple interactive (non-batch) jobs
  - Starting point for Unix operating systems in the 1960‘s
- Preemptive multi-tasking became a single user demand in modern times
  - Leave application running while starting another one
- Pure batch processing systems are still significant (TPM, SAP R/3, HPC)

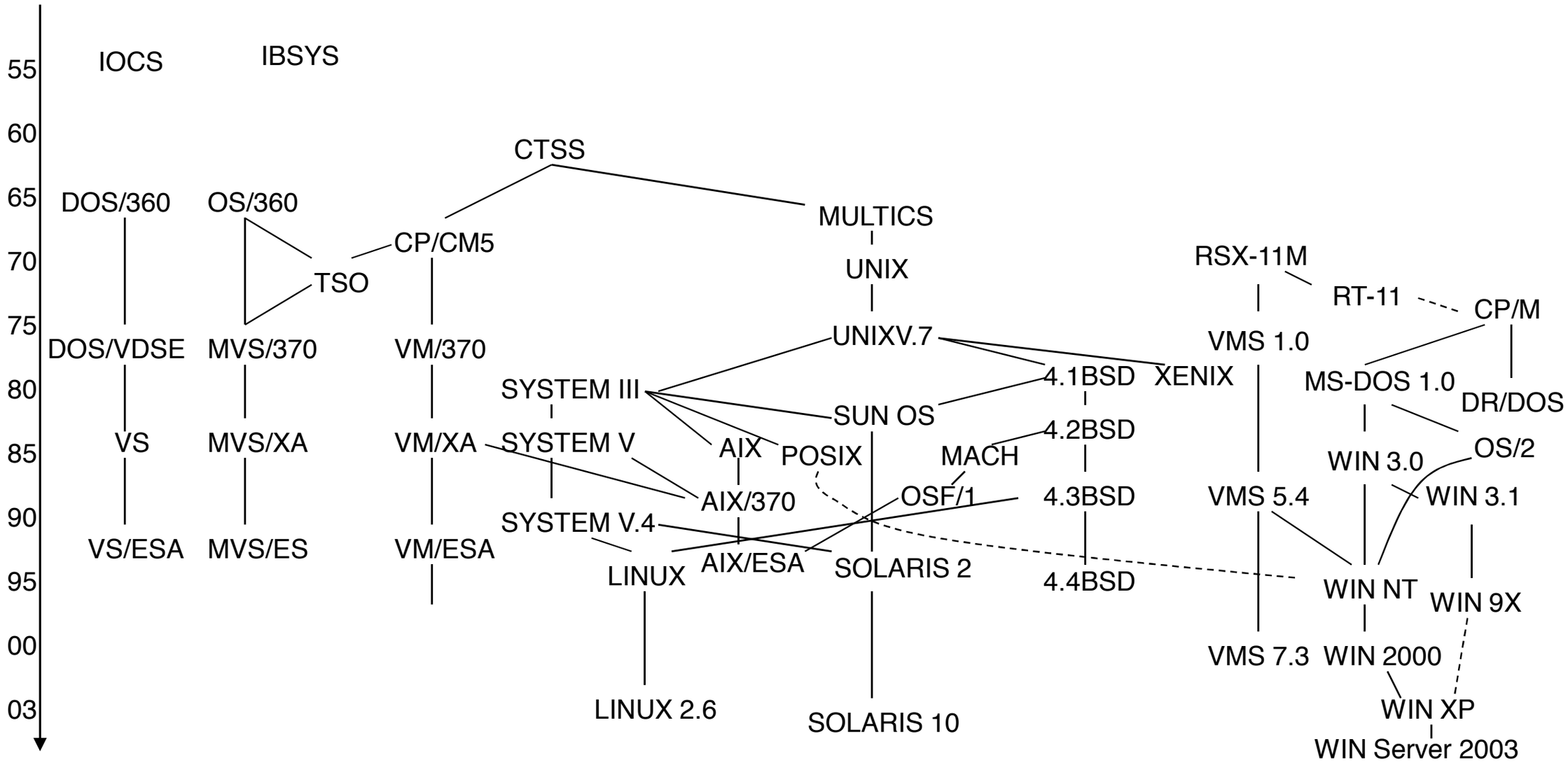
Time-Sharing

- Compatible Time-Sharing System (**CTSS**)
  - Operating system developed at MIT, first for the IBM 7094 in 1961 (32.768 36bit words memory)
  - Program always loaded to start at the location of the 5000th word
- System clock generated interrupts roughly every 0.2 seconds



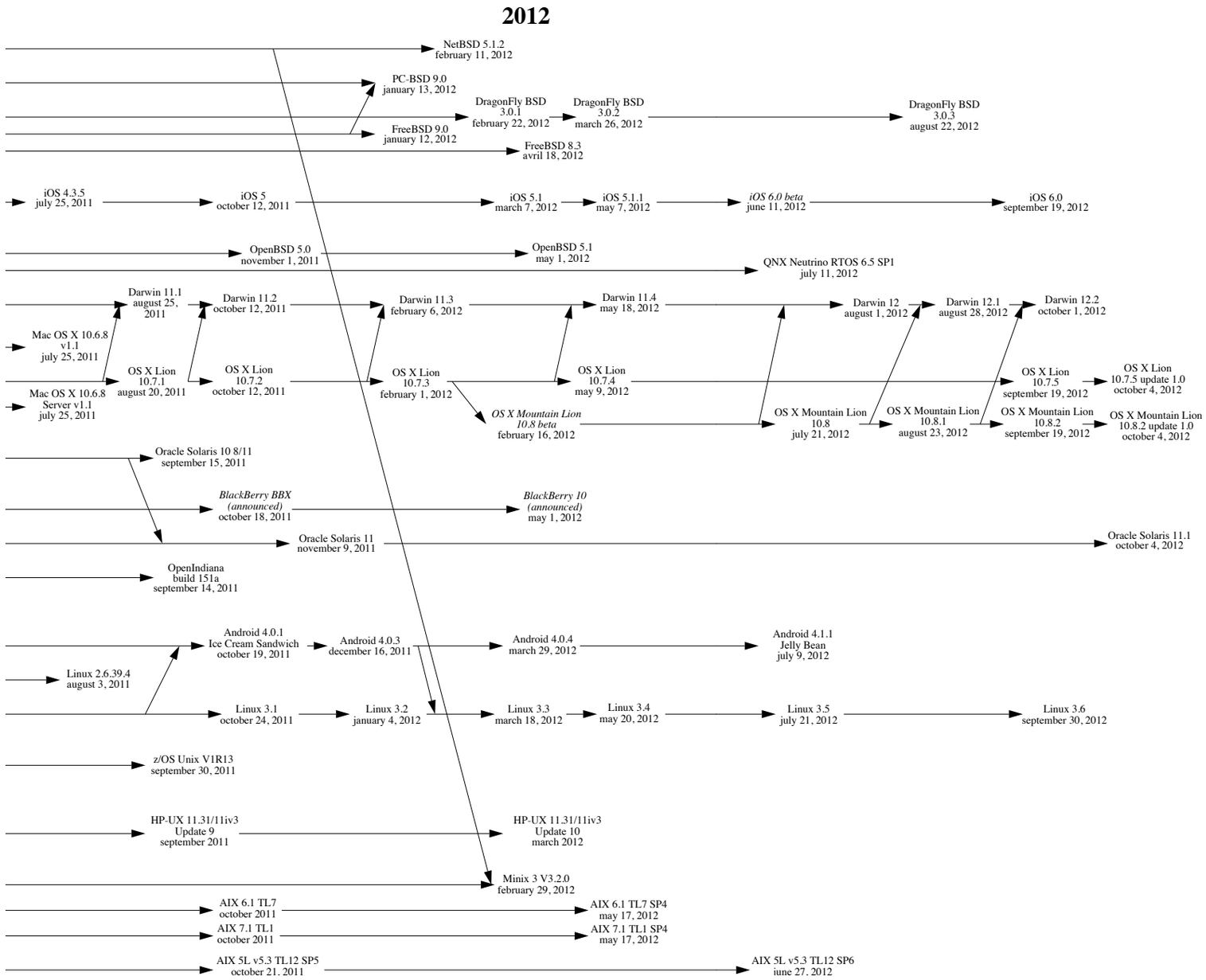
- At each clock interrupt, the system regained control and assigned the processor to another user - **time slicing**
- Parts of the active program that would be overwritten are written to disk
  - Other parts remained inactive in the system memory
- Direct successor MULTICS pioneered many modern operating system concepts

History of Modern Operating Systems



History of Unix [levenez.com]

- Compatible Time-Sharing System (CTSS)
  - 1961-1973, MIT
  - First time-sharing system
  - Successor MULTICS
- September 1969: UNICS
- UNIX as today
  - Originally from Bell Labs
  - BSD Unix derivate by University of Berkeley



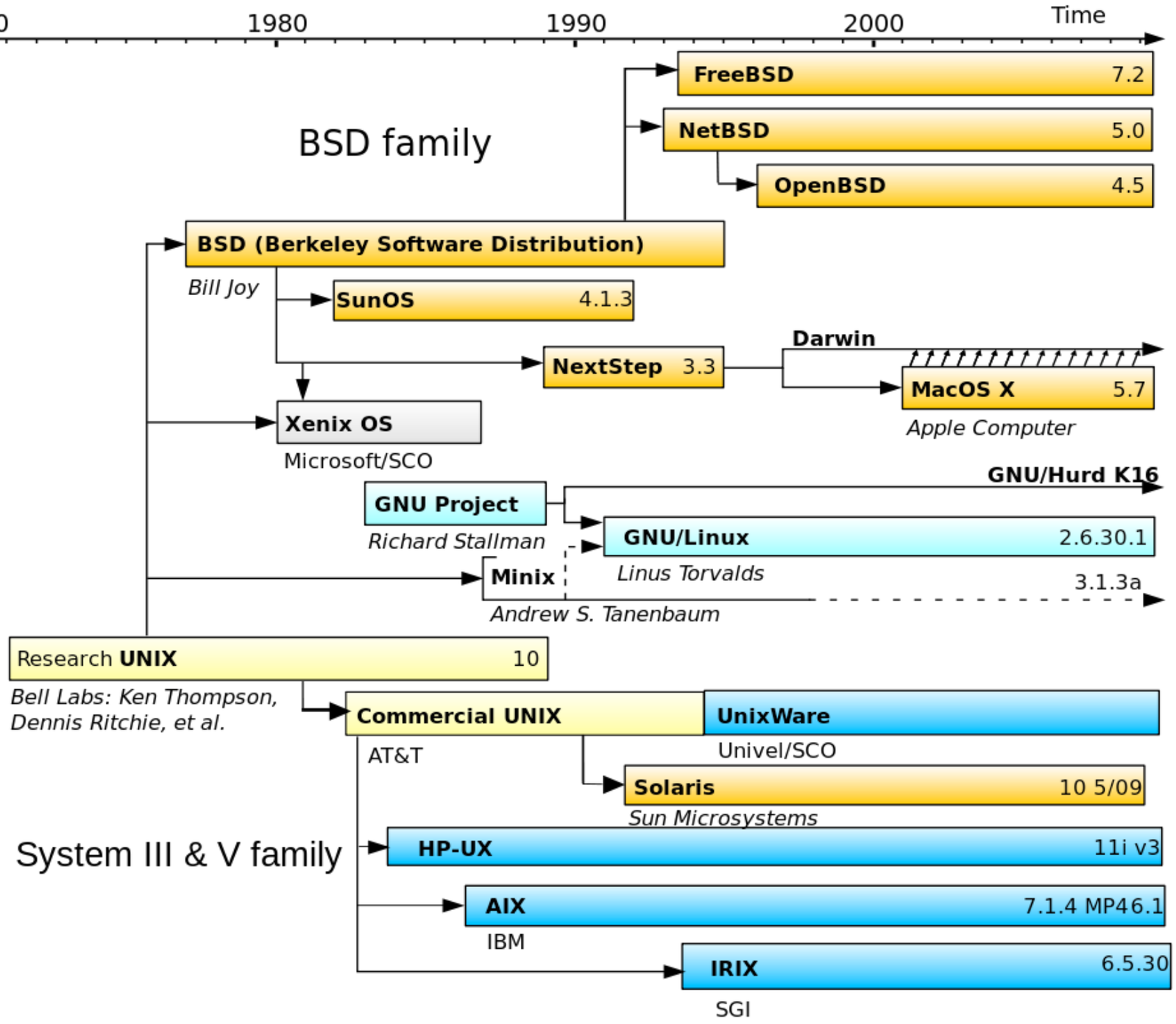
History of Unix

- First version of UNIX from Bell Labs in 1969 was influenced by CTSS and MULTICS
  - Developed for PDP-7
  - Hierarchical file system, processes, device files, command-line interpreter, tools
- Re-write from assembly language to C in 1973
- First widely available version 6 in 1976
- Bill Joy created the first Berkeley University Distribution (BSD) for UnixV6 in 1978
  - Initial porting of Unix sources to VAX computers
  - Release of **vi** editor and **C shell** with second version in 1979
  - Many BSD contributions were adopted by Unix and others, especially the TCP/IP stack implementation and Berkeley socket API
- Unix Version 7 (1978) is the conceptional ancestor of most modern UNIX systems

History of Unix

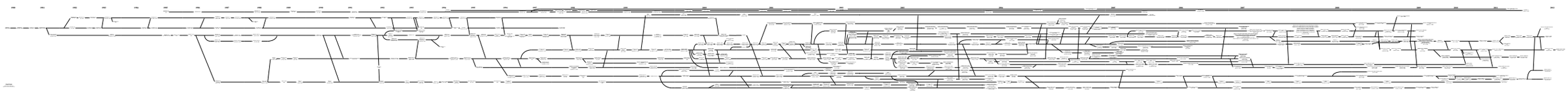
- 1983: AT&T UNIX System V released as one of the first commercial versions
  - Interface definition still used for modern Unix systems such as AIX and Solaris
  - Richard Stallman starts the GNU project for free Unix-compatible software
- 1988: Different APIs ultimately united by **IEEE POSIX** specification
- 1989: **SVR4** unification of BSD and System V / Windows NT development starts
- 1991: Linus Torvalds begins to work an a Unix clone for IBM PCs - **Linux**
- 1992: Berkeley alumni‘s publish **386BSD** port of BSD to Intel‘s 386, foundation for later **FreeBSD** and **NetBSD** unix versions
- Since 1996: Unix trademark owned by **Open Group**, certification process
- 1997: Apple creates **Darwin** kernel out of the Mach kernel and Unix BSD parts
- 1999: System V release 4 (SVR4) binary format ELF became agreed Unix standard

History of Unix



from Wikipedia

History of Windows [levenez.com]



- August, 1980: QDOS 0.1
- April 6th, 1992: Windows 3.1
- July 27th, 1993: Windows NT 3.1
- August 24th, 1995: Windows 95
- August 24th, 1996: Windows NT 4.0
- September 24th, 2001: Windows XP
- September 4th, 2012: Windows Server 2012
- ... Windows 8 ...

History of Windows [Lucovsky]

- Initial team formed in November 1988
  - 6 former Digital developers, one Microsoft guy
  - Focus on secure, scalable SMP design for desktops and servers
  - Original schedule for 18 months, missed by 3 years
- Goal setting
  - Portability: Initial focus on Intel i860, intentionally late focus on i386
  - Reliability: Nothing should be able to crash the OS (promise fulfilled)
  - Extensibility and compatibility (DOS, OS/2, POSIX)
  - After all of the above - performance
- NT 3.1 had 6-200 developers, NT 4.0 800 developers, Windows 2000 had 1400

Windows and Linux Evolution

- Windows and Linux kernels are based on foundations developed in the mid-1970s