6.823
Computer System Architecture

http://www.csg.lcs.mit.edu/6.823

The emphasis in this course is on architectural mechanisms and not on quantitative performance evaluation.

Lecturer: ***Krste Asanovic***
TA's: ***Mike Sung (Head TA), Seongmoo Heo, Albert Ma, Michael Zhang***

Course Information

http://www.csg.lcs.mit.edu/6.823

You must sign up for the course through the web

~7 Homeworks (30%) ***Midterm (30%)*** ***Final (40%)***

Tutorials on Fridays, starting Feb 9

- **11:00-12:00 in room 31-161**
- **2:00-3:00 in room 37-212**

All students must help grade homeworks once during semester, signup sheets distributed during class

Problem Set 0

- Goal is to help you judge for yourself whether you have prerequisites for this class
- We assume that you understand digital logic, a simple 5-stage pipeline, and simple caches
- For this problem set only, work by yourself – not in groups
- Remember to complete self-evaluation section at end of Problem Set 0
- Due at start of class Monday Feb 12

Early Developments: From *Difference Engine* to *IBM 701*

Daniel L. Rosenband
(danlief@lcs.mit.edu)
Laboratory for Computer Science
M.I.T.

http://www.csg.lcs.mit.edu/6.823

Charles Babbage 1791-1871

Lucasian Professor of Mathematics
Cambridge University, 1827-1839

Difference Engine **1823**

Analytic Engine **1833**
The forerunner of modern digital computer!

Application?
Mathematical Tables - Astronomy
Nautical Tables - Navy

Background
Any continuous function can be approximated by a polynomial --- *Weierstrass*

Technology
mechanical - gears, Jacquard's loom, simple calculators

Difference Engine

A machine to compute mathematical tables

Weierstrass:
- Any continuous function can be approximated by a polynomial
- Any Polynomial can be computed from *difference* tables

An example,

$f(n) = n^2+n+41$

$d1(n) = f(n) - f(n-1) = 2n$

$d2(n) = d1(n) - d1(n-1) = 2$

$f(n) = f(n-1) + d1(n) = f(n-1) + (d1(n-1) + 2)$

n	0	1	2	3	4 ...
d2(n)			2	2	2
d1(n)		2	4		
f(n)	41	43	47		

all you need is an adder!

Difference Engine

1823 - Babbage's paper is published
1834 - The paper is read by Scheutz & his son in Sweden
1842 - Babbage gives up the idea of building it; (he is onto Analytic Engine!)
1855 - Scheutz displays his machine at the Paris World Fare
- Can compute any 6th degree polynomial
- *Speed:* 33 to 44 32-digit numbers per minute!

Now the machine is at the Smithsonian

Analytic Engine

1833 - Babbage's paper is published
conceived during a hiatus in the development of the difference engine

Inspiration:*Jacquard Looms*
looms controlled by punched cards

The set of cards with fixed punched holes dictated the pattern of weave $\Rightarrow$ *program*
The same set of cards could be used with different colored threads $\Rightarrow$ *numbers*

1871 - Babbage dies - the machine remains unrealized.
It is not clear if the analytic engine could be built even today using only mechanical technology

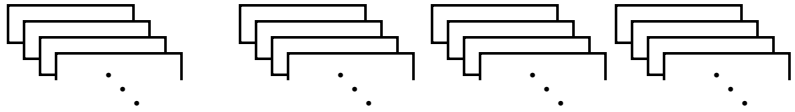
Analytic Engine

The first conception of a general purpose computer

1. **The *store* in which all variables to be operated upon, as well as all those quantities which have arisen from the results of the operations are placed.**
2. **The *mill* into which the quantities about to be operated upon are always brought.**

The *program*

Operation **variable1** **variable2** **variable3**



An operation in the *mill* required feeding two punched cards and producing a new punched card for the *store.*

An operation to alter the sequence was also provided!

The first programmer

Ada Byron *aka* "Lady Lovelace"

Babbage's ideas had a lot of influence later, primarily because of

- ***Luigi Menabrea*, who published notes of Babbage's lectures in Italy**
- ***Lady Lovelace*, who translated Menabrea's notes in English and thoroughly expanded them.**
  "... Analytic Engine weaves *algebraic patterns*...."

Ada's tutor was Babbage himself!

In the early twentieth century - the focus shifted to analog computers but

Harvard Mark I built in 1944 is very close in spirit to the Analytic Engine.

Linear Equation Solver

John Atanasoff, Iowa State University

1930's: Atanasoff built the Linear Equation Solver.
It had 300 tubes!

Application:
Linear and Integral differential equations

Background:
Vannevar Bush's Differential Analyzer
--- *an analog computer*

Technology:
Tubes and Electromechanical relays

Atanasoff decided that the correct mode of computation was by electronic digital means.

ENIAC and EDVAC

The first conception of a *stored program computer*

ENIAC 1946, 48
EDVAC 1948 ***concept only***

Players brought together by the WW-2 effort
- Eckert & Mauchley, University of Pennsylvania
- John von Neumann, Princeton University

Application:
Ballistic calculations

angle = f (location, tail wind, cross wind,
air density, temperature, weight of shell,
propellant charge, ...)

Technology:
tubes, relays, electromechanical delays,
mercury delay lines,...

Developed the concept of *stored program computer*
⇒ *program can be manipulated as data*

Stored Program Computer

Program = A sequence of instructions

How to control instruction sequencing?

manual control	**calculators**
automatic control	
external (paper tape)	**Harvard Mark I , 1944**
	Zuse's Z1, WW2
internal	
plug board	**ENIAC 1946**
read-only memory	**ENIAC 1948**
read-write memory	**EDVAC 1947 *(concept)***

⇒ The same storage can be used to store program and data

EDSAC	**1950**	**Maurice Wilkes**

Technology Issues

ENIAC	**⇒**	**EDVAC**
18,000 tubes		**4,000 tubes**
20 10-digit numbers		**2000 word storage mercury delay lines**

ENIAC had many asynchronous parallel units but only one was active at a time

BINAC : Two processors that checked each other for reliability.

Didn't work well because processors never agreed

The Spread of Ideas

ENIAC & EDVAC had immediate impact
brilliant engineering: **Eckert & Mauchley**
lucid paper: **Burks, Goldstein & von Neumann**

IAS	**Princeton**	**46-52 Bigelow**
EDSAC	**Cambridge**	**46-50 Wilkes**
MANIAC	**Los Alamos**	**49-52 Metropolis**
JOHNIAC	**Rand**	**50-53**
ILLIAC	**Illinois**	**49-52**
	Argonne	**49-53**
SWAC	**UCLA-NBS**	

UNIVAC - the first commercial computer, 1951

Alan Turing's direct influence on these developments is still being debated by historians.

Dominant Problem: *Reliability*

Mean time between failures (MTBF)
MIT's Whirlwind with an MTBF of 20 min. was perhaps the most reliable machine !

Reasons for unreliability:

1. **Vacuum Tubes**

2. **Storage medium**
   - **acoustic delay lines**
   - **mercury delay lines**
   - **Williams tubes**
   - **Selections**

CORE J. Forrester 1954

Commercial Activity -- 1948-52

IBM's SSEC

Selective Sequence Electronic Calculator

- **150 word store.**
- **Instructions, constraints, and tables of data were read from paper tapes.**
- **66 Tape reading stations!**
- **Tapes could be glued together to form a loop!**
- **Data could be output in one phase of computation and read in the next phase of computation.**

And then there was IBM 701

IBM 701 -- 30 machines were sold in 1953-54

IBM 650 -- a cheaper, drum based machine, more than 120 were sold in 1954 and there were orders for 750 more!

Users stopped building their own machines.

Why was IBM late getting into computer technology?

They were making too much money!

Even without computers, IBM revenues were doubling every 4 to 5 years in 40's and 50's.

Software Developments

up to 1955 **Libraries of numerical routines**
- **Floating point operations**
- **Transcendental functions**
- **Matrix manipulation, equation solvers, . . .**

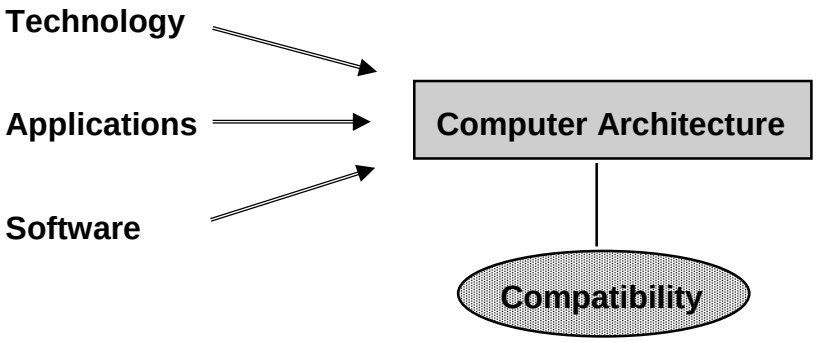
1955-60 ***High level Languages* - Fortran 1956**
***Operating Systems* -**
- **Assemblers, Loaders, Linkers, Compilers**
- **Accounting programs to keep track of usage and charges**

Machines required *experienced operators*

⇒ Most users could not be expected to understand these programs, much less write them

⇒ Machines had to be sold with a lot of resident software

Factors that Influence Computer Architecture



Software played almost no role in defining an architecture before mid fifties.

special-purpose *versus* general-purpose machines

Microprocessors Economics

In mid nineties

- **Designing a state-of-the-art microprocessor requires a huge design team**
  - **Pentium ~300 engineers**
  - **PentiumPro ~ 500 engineers**
- **Huge investments in fabrication lines**
  - **⇒ need to sell 2 to 4 million units to be profitable**
- **Continuous improvements are needed to improve yeilds and clock speed**
  - **⇒ price drops to one tenth in 2-3 years**
- **Fast new processors also require new peripheral chips (memory controller, I/O) ⇒ $$$**
- ***Cost of launching a new ISA is prohibitive and the advantage is not so clear!***

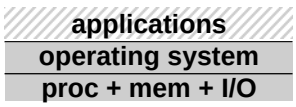
Compatibility

Essential for *portability* and *competition*
- **Its importance increases with the market size**
- **but it is also the most *regressive* force**

What does compatibility mean?

Instruction Set Architecture (ISA) compatibility
- **The same assembly program can run on any upward compatible model**
- ***then* IBM 360/370 ... *now* Intel x86**

System and application software developers expect more than ISA compatibility (API's)



Java?

Wintel

Technology-Driven Views of Computer Architecture - 1

Implement an (old) ISA in new technology using new micro-architectures

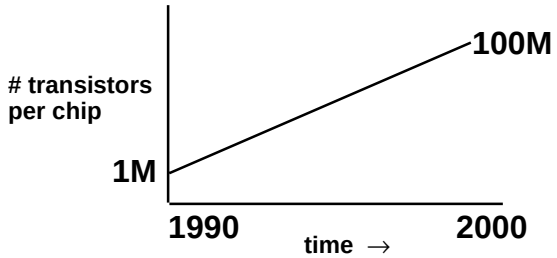
An iterative process

- **select some new features**
- **design datapaths and control**
- **estimate cost**
- **measure performance on simulators**

Sought after expertise

- ***Understanding of micro-architectures***
- ***Hardware and circuit design***
- ***Simulation, verification & testing***
- ***Back-end compilers and performance evaluation***

Technology-Driven Views of Computer Architecture - 2



What exciting things can we do with the latest technology.

Consider a 64-bit RISC processor,1997 vintage:
4-way superscaler, I & D L1 caches, L2 controller
out-of-order and speculative execution,... ***6M trans.***

Will future billion transister chip contain
more of the same (i.e., mostly caches)?
multiprocessors (i.e., SMP's sans DRAM)?
full systems = processor(s)+DRAM+I/O?

Language-Driven View of Computer Architecture

Since software is the most costly activity, design machines to support high-level languages

Language designers consider *abstractions* for
- **expressiveness**
- **enhanced programmability**
- **portability**
- **insulation from small changes in hardware**

Architects provide *hardware mechanisms* to support these abstractions in a cost effective manner

The field is littered with some instructive and spectacular *failures* of language-driven architectures
e.g., Stack machines, Lisp Machines, ...

My view

Language/ Compiler/ System software designer		***Architect/Hardware designer***
Need mechanisms to support important abstractions	$\Rightarrow$	**Decompose each mechanism into essential micro-mechanisms and determine its feasibility and cost effectiveness**
Determine compilation strategy; new language abstractions	$\Leftarrow$	**Propose mechanisms and features for performance**

Architects main concerns are cost-performance , performance, and efficiency in supporting a broad class of software systems.