Computing Devices Now
Computing Devices Then...
A Journey Through Space

• What is Computer architecture and what do computer architects actually do?

• Illustrate via historical examples
  ‣ Prehistory: Babbage and Analytic Engine
  ‣ Early days: Eniac, Edvac and Edsac
  ‣ Arrival of IBM 650 and then IBM 360
  ‣ Seymour Cray – CDC 6600, Cray 1
  ‣ Microprocessors, Multicores, SoCs

• Focus on ideas and mechanisms that have withstood the test of time
• Design of the abstraction layers

Original domain of the computer architect (‘50s–‘80s)

Application
Algorithm
Programming Language
Operating System/Virtual Machine
Instruction Set Architecture (ISA)
Microarchitecture
Register-Transfer Level (RTL)
Circuits
Devices
Physics

Parallel computing security, ...

Domain of computer architecture (‘90s)
Reliability, power
Importance of Technology

• New technologies not only provide greater speed, size and reliability at lower cost, but more importantly these dictate the kinds of structures that can be considered and thus come to shape our whole view of what a computer is.

Bell & Newell
• Technology is the dominant factor in computer design

Technology
Transistors
Integrated circuits
VLSI (initially)

Technology
Core memories
Magnetic tapes
Disks

Technology
ROMs, RAMs
VLSI
Packaging
Low Power
But Software…

• As people write programs and use computers, our understanding of *programming* and *program behavior* improves.
  ‣ *This has profound though slower impact on computer architecture*

• Modern architects cannot avoid paying attention to software and compilation issues.

Software → Computers ← Technology
• Architecture is Engineering Design under constraints
  ‣ Factors to consider:
    • Performance of whole system on target applications
      ‣ Average case & worst case
    • Cost of manufacturing chips and supporting system
    • Power to run system
      ‣ Peak power & energy per operation
    • Reliability of system
      ‣ Soft errors & hard errors
Engineering Design

• Architecture is Engineering Design under constraints

  ‣ Factors to consider:
    • Cost to design chips (engineers, computers, CAD tools)
      ‣ Becoming a limiting factor in many situations, fewer unique chips can be justified
    • Cost to develop applications and system software
      ‣ Often the dominant constraint for any programmable device
Charles Babbage 1791–1871

- **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
  - Any Polynomial can be computed from difference tables
The first programmer

• Ada Byron aka “Lady Lovelace” 1815-52

• Ada’s tutor was Babbage himself!
Harvard Mark I

- Built in 1944 in IBM Endicott laboratories
  - Howard Aiken – Professor of Physics at Harvard
  - Essentially mechanical but had some electro-magnetically controlled relays and gears
    - Weighed 5 tons and had 750,000 components
    - A synchronizing clock that beat every 0.015 seconds

- Performance:
  - 0.3 seconds for addition
  - 6 seconds for multiplication
  - 1 minute for a sine calculation

Broke down once a week!
Linear Equation Solver

• 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
Electronic Numerical Integrator

- Designed and built by Eckert and Mauchly at the University of Pennsylvania during 1943-45
- The first, completely electronic, operational, general-purpose analytical calculator!
  - 30 tons, 72 square meters, 200KW
- Performance
  - Read in 120 cards per minute
  - Addition took 200 ms, Division 6 ms
  - 1000 times faster than Mark I
- Not very reliable!
Automatic Computer

- Electronic Discrete Variable Automatic Computer
- ENIAC’s programming system was external
  - Sequences of instructions were executed independently of the results of the calculation
  - Human intervention required to take instructions “out of order”
- EDVAC was designed by Eckert, Mauchly and von Neumann in 1944 to solve this problem
  - Solution was the stored program computer
  - “program can be manipulated as data”
• First Draft of a report on EDVAC was published in 1945, but just had von Neumann’s signature!

• Without a doubt the most influential paper in computer architecture
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
Program = A sequence of instructions

- How to control instruction sequencing?
  - Internal
    - read-only memory  ENIAC  1948
    - read-write memory  EDVAC  1947 (concept)
    - EDSAC  1950  Maurice Wilkes
      - considered the first complete and fully operational regular electronic digital stored-program computer
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
• UNIVAC - the first commercial computer, 1951
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
Architecture Features

• HD Boost.
  ‣ significant gains on the latest SSE4 instruction set

• Hyper-Threading Technology.
  ‣ each core processes two application “threads” simultaneously

• Turbo Boost Technology.
  ‣ increases the processor’s frequency when needed

• True quad-core
  ‣ enables cores to communicate at die level
Architecture Features

• Wide Dynamic Execution
  ‣ enables the delivery of more instructions per clock cycle

• Intelligent Power Capability
  ‣ turning off portions of the processor when they aren't being used

• Smart Memory Access
  ‣ increasing available data bandwidth

• 8 MB Shared Smart Cache.
  ‣ enabling multiple cores to dynamically share this space
The course has 4 modules

**Module 1**
- Instruction Set Architecture (ISA)
- Simple Pipelining and Hazards

**Module 2**
- Superscalar Architectures
- Superscalar Processors
- Vector machines
- VLIW
- Multithreading
- GPU

**Module 3**
- Branch Prediction
- Caches
- Memory Models & Synchronization
- Cache Coherence Protocols

**Module 4**
- On-Chip networks
- On-chip Network routing
- Reliability
Acknowledgments

• Class slides contain material developed and/or taught by:
  • Prof. Arvind (MIT)
  • Prof. Srinivas Devadas (MIT)
  • Prof. Krste Asanovic (MIT/UCB)
  • Prof. & Senior Intel Fellow Joel Emer (MIT/Intel)
  • Daniel Sanchez (MIT)
• And all the teaching assistants of MIT 6.823  
  ‣ A group that I’m fortunate enough to call myself a member