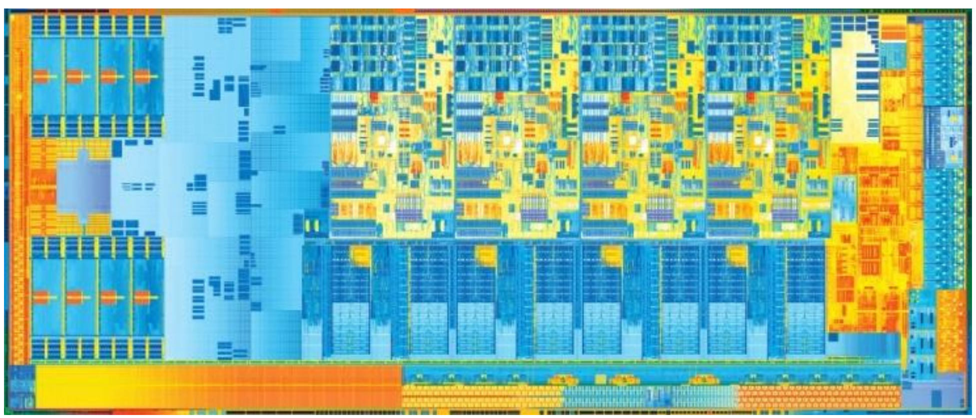
6.823 Computer System Architecture

Lecturers: *Daniel Sanchez and Joel Emer*

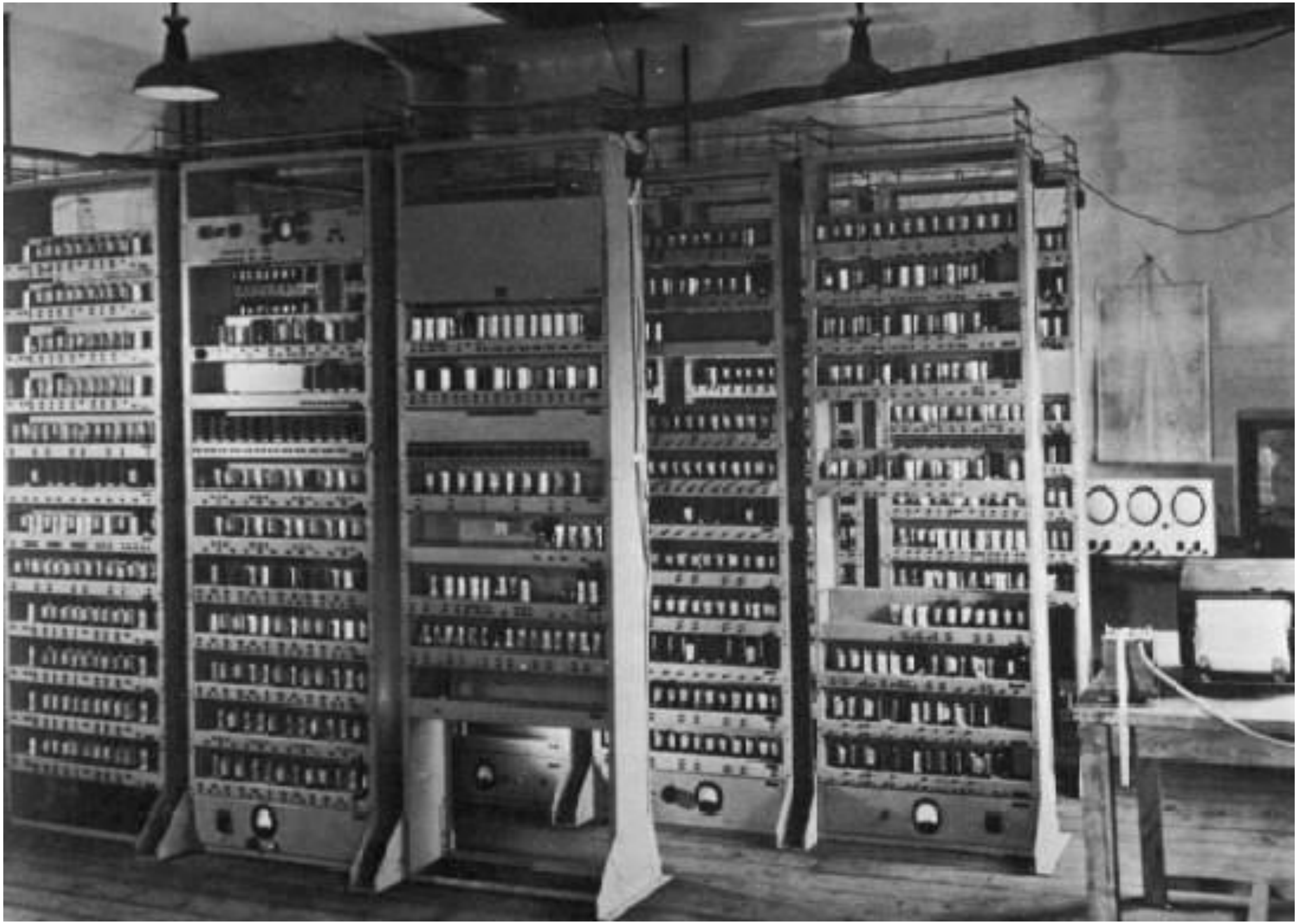
TA: *Nathan Beckmann*

The processor you built in 6.004*



What you'll understand after taking 6.823

Computing Devices Then…



Computing Devices Now

Some marketing buzzwords

- Wide Dynamic Execution
  - Enables the delivery of more instructions per clock cycle
- Hyper-Threading Technology
  - Each core processes two application “threads” simultaneously
- HD Boost
  - Significant gains on the latest SSE4 instruction set
- Turbo Boost Technology
  - Increases the processor’s frequency when needed
- Many Integrated Core (MIC) Architecture
  - Many cores on a single chip
- 8 MB Shared Smart Cache
  - Enables multiple cores to dynamically share cache space
- Smart Memory Access
  - Increases available data bandwidth
- Intelligent Power Capability
  - Turns off portions of the processor when they aren't being used

A journey through this space

- What do computer architects actually do?

A journey through this space

- 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 and PCs
  - Multicores
  - Cell phones

A journey through this space

- What do computer architects actually do?

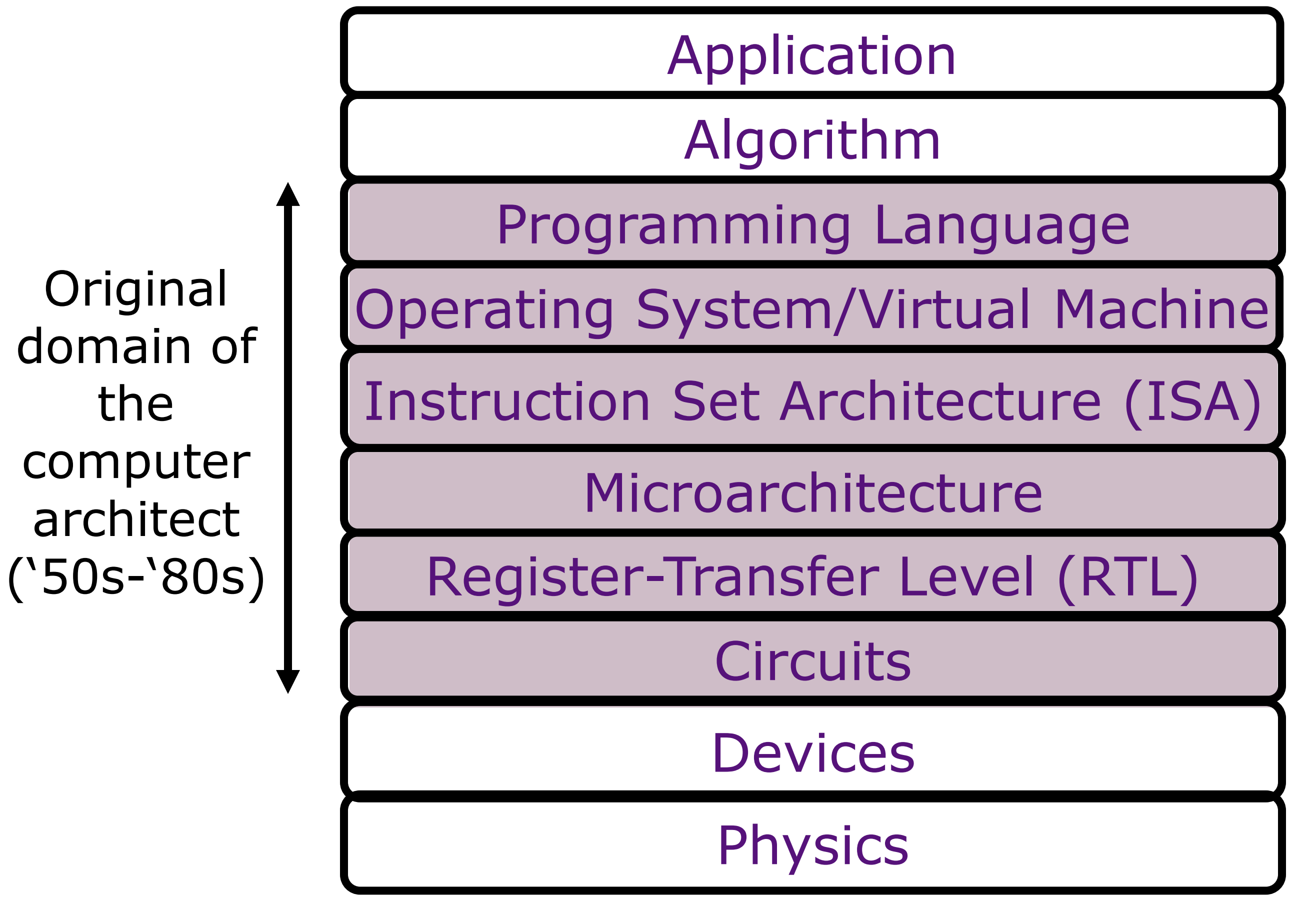
- Illustrate via historical examples
  - Prehistory: Babbage and Analytic Engine
  - Early days: Eniac, Edvac and Edsac
  - Arrival of IBM 650 and then IBM 360
  - Seymour Cray – CDC 6600, Cray 1
  - Microprocessors and PCs
  - Multicores
  - Cell phones

- Focus on ideas, mechanisms and principles, especially those that have withstood the test of time

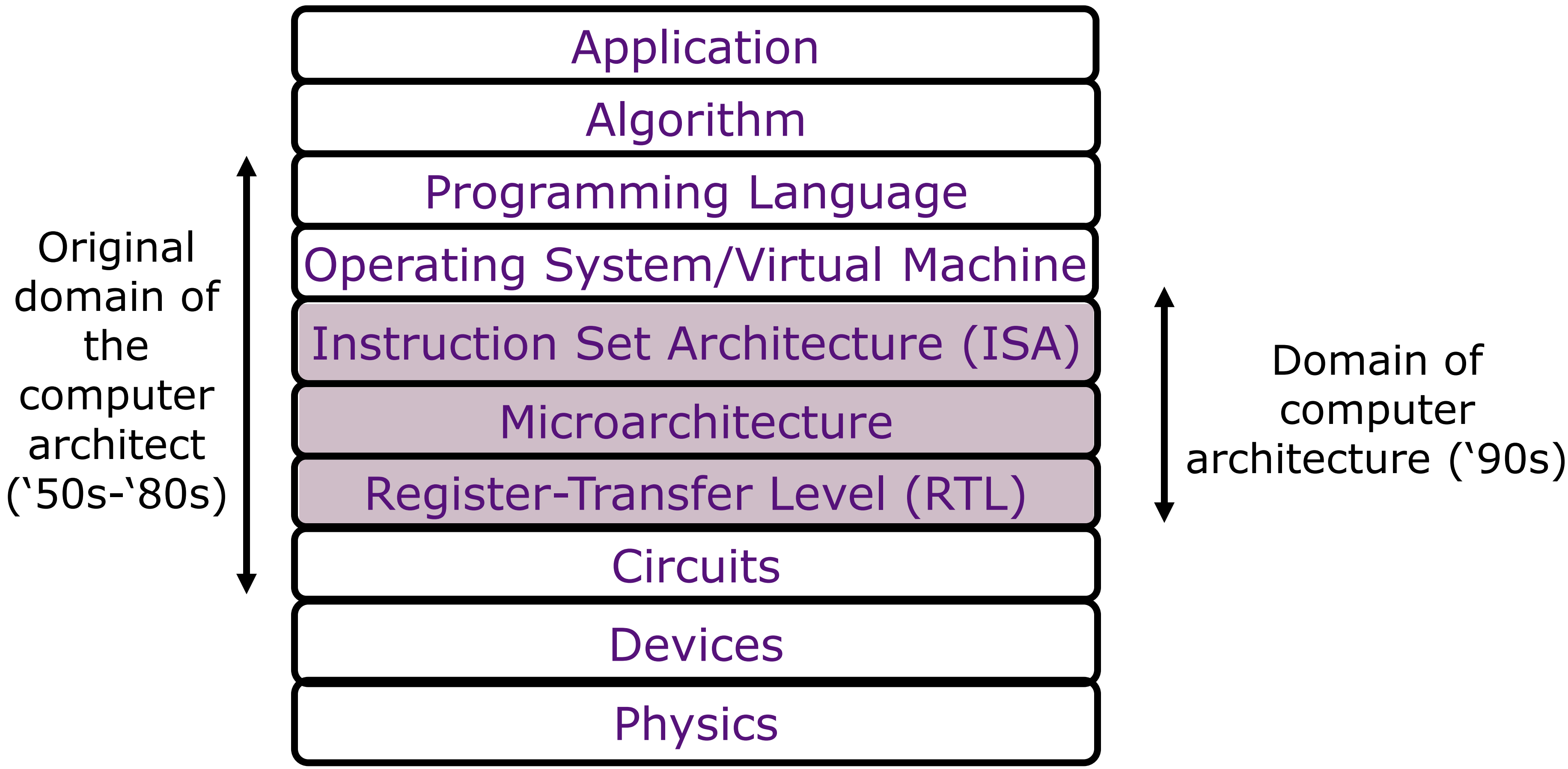
Abstraction Layers

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

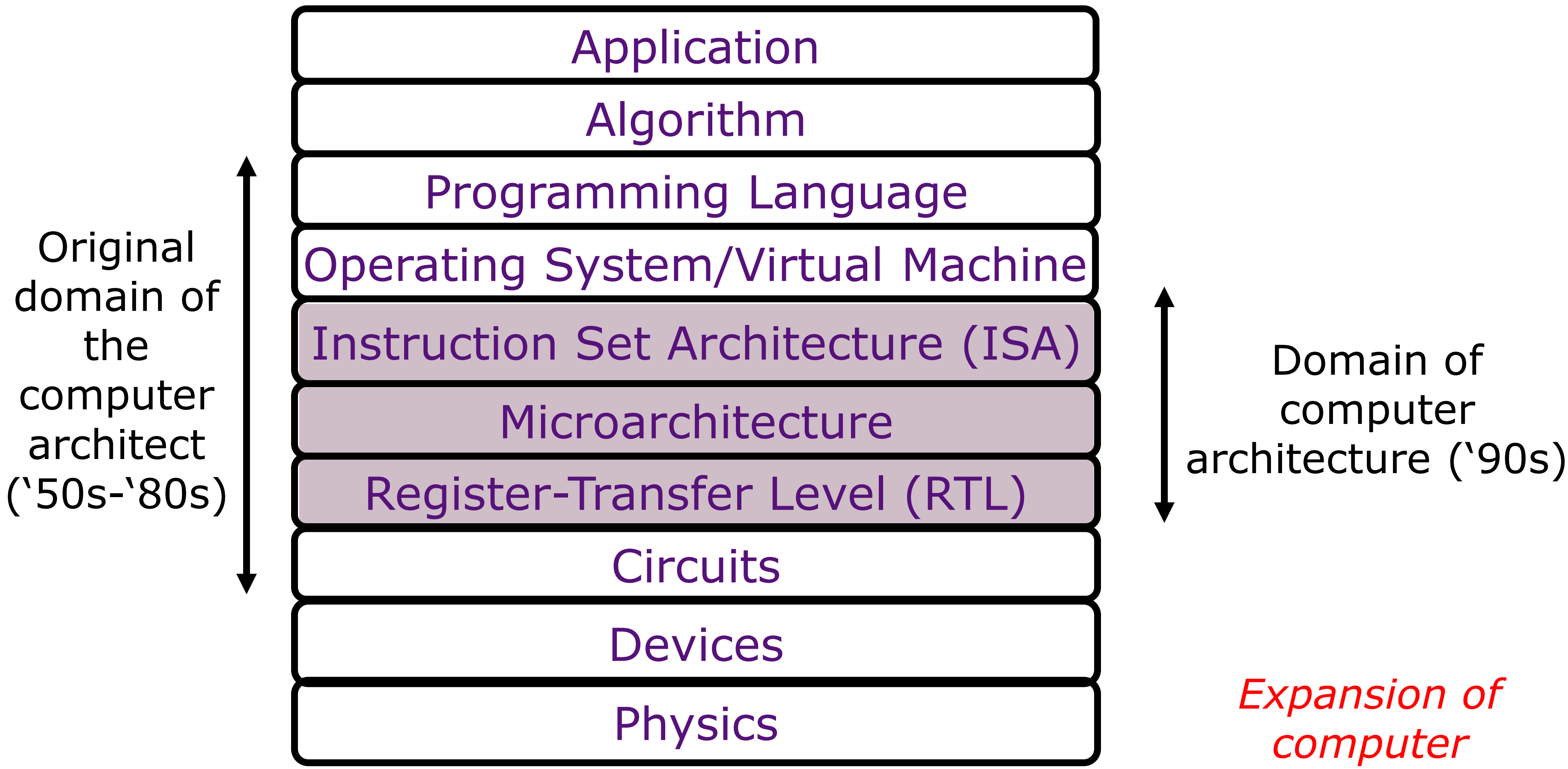
Abstraction Layers



Abstraction Layers

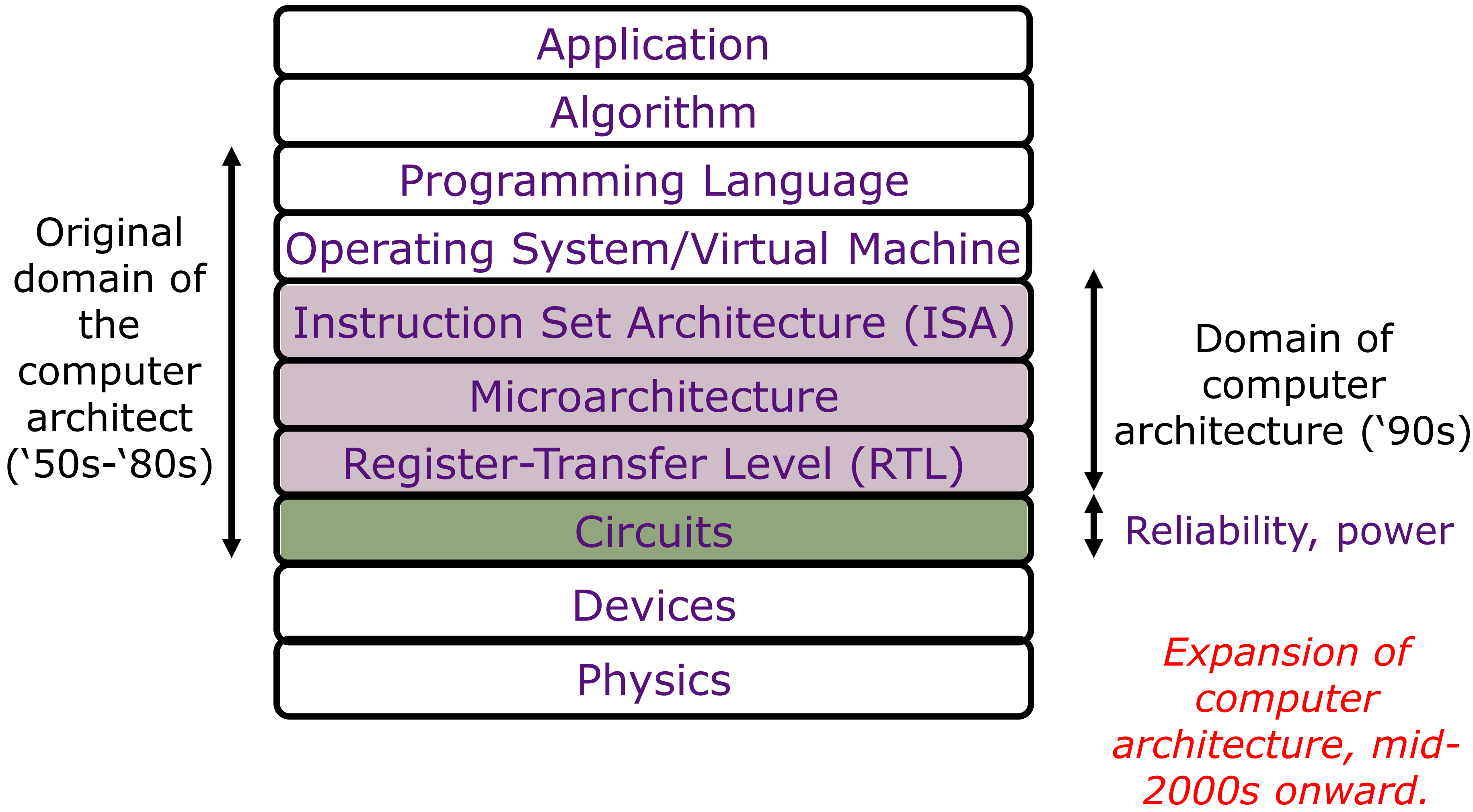


Abstraction Layers

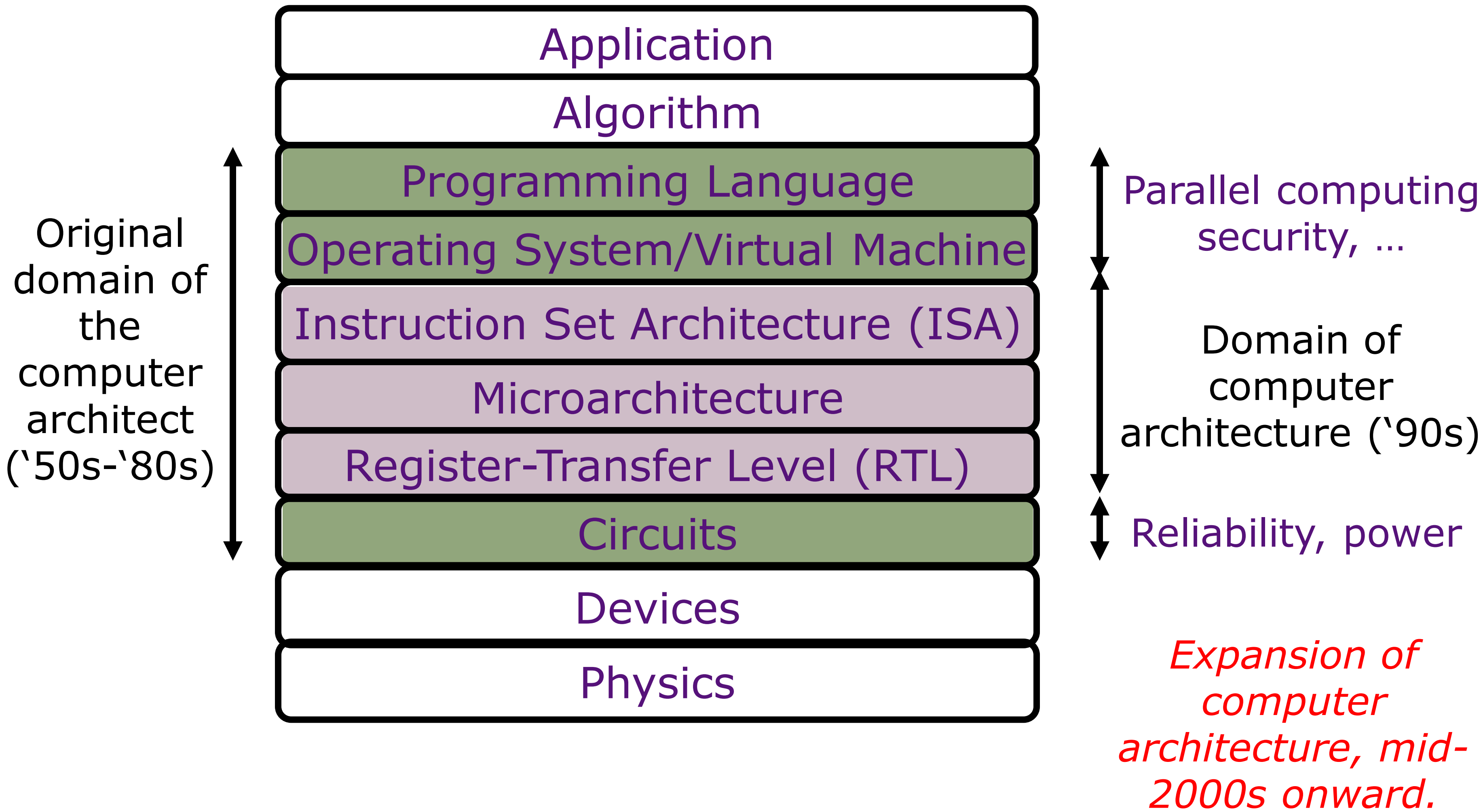


Expansion of computer architecture, mid-2000s onward.

Abstraction Layers



Abstraction Layers



Computer Architecture is the design of abstraction layers

Computer Architecture is the design of abstraction layers

- What do abstraction layers provide?
  - Environmental stability within generation
  - Environmental stability across generations
  - Consistency across a large number of units

Computer Architecture is the design of abstraction layers

- What do abstraction layers provide?
  - Environmental stability within generation
  - Environmental stability across generations
  - Consistency across a large number of units

- What are the consequences?
  - *Encouragement to create reusable foundations:*
    - *Tool chains, operating systems, libraries*
  - Enticement for application innovation

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.

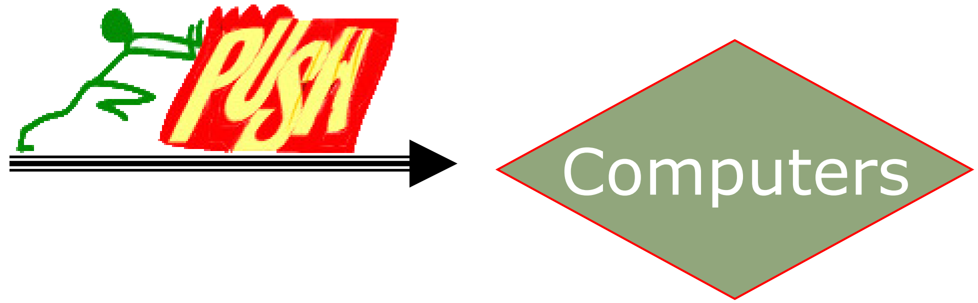
Bell & Newell

Technology is the dominant factor in computer design

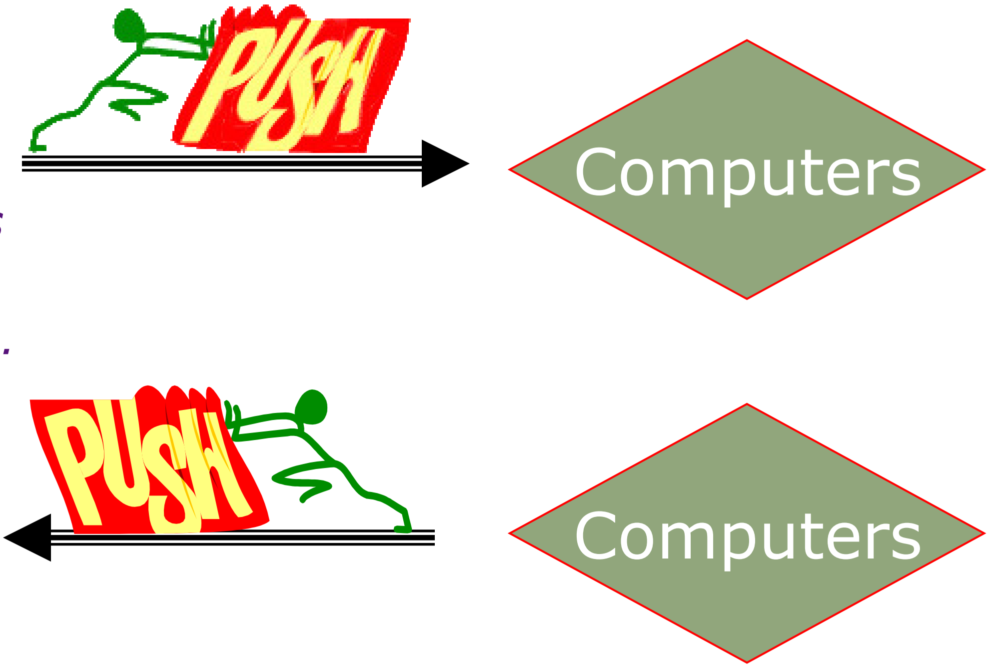
Technology is the dominant factor in computer design

Technology

Transistors
Integrated circuits
VLSI (initially)
Flash memories, …



Technology is the dominant factor in computer design



Technology

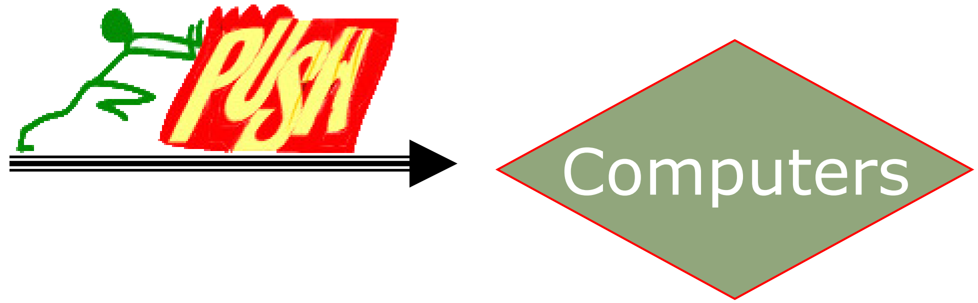
Transistors
Integrated circuits
VLSI (initially)
Flash memories, …

Technology

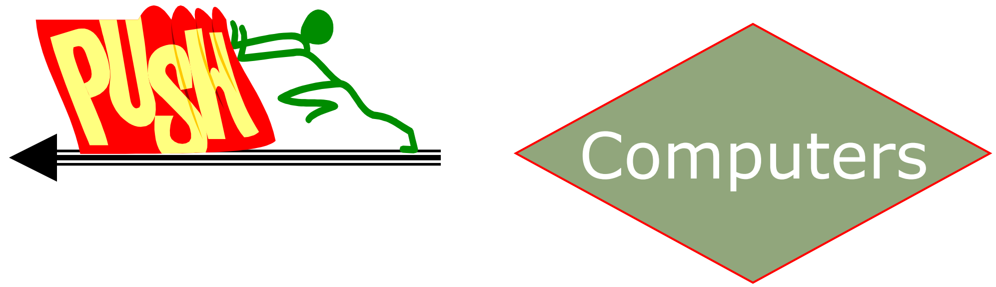
Core memories
Magnetic tapes
Disks

Technology is the dominant factor in computer design

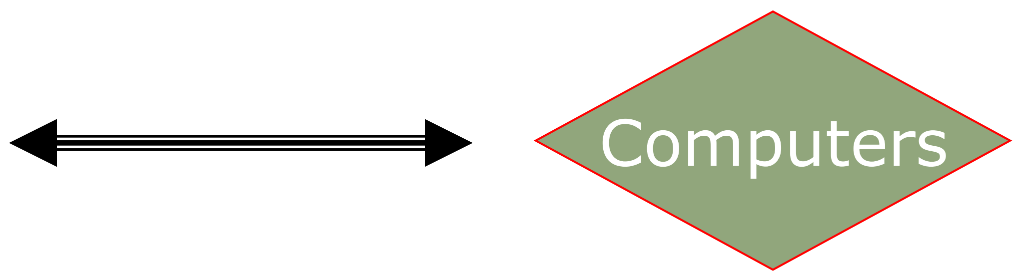
Technology
- *Transistors*
- *Integrated circuits*
- *VLSI (initially)*
- *Flash memories, …*



Technology
- *Core memories*
- *Magnetic tapes*
- *Disks*



Technology
- *ROMs, RAMs*
- *VLSI*
- *Packaging*
- *Low Power*



But Software...

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

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

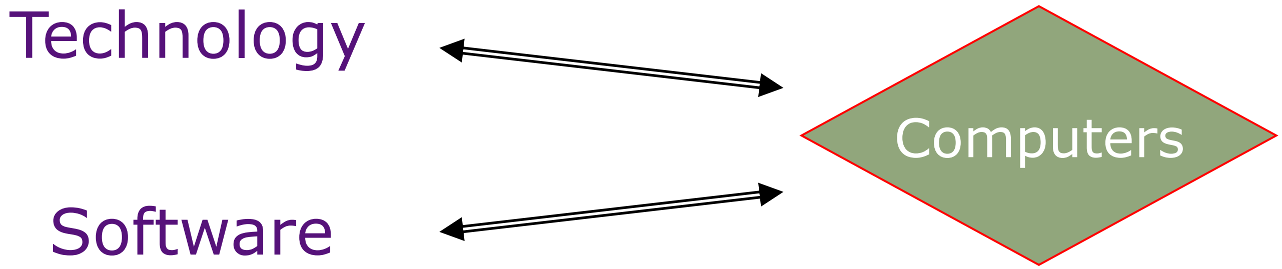
Modern architects cannot avoid paying attention to software and compilation issues.

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.



Architecture is Engineering Design under Constraints

Factors to consider:

Architecture is Engineering Design under Constraints

Factors to consider:

- Performance of whole system on target applications
  - Average case & worst case

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

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

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

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
- Cost to design chips (engineers, computers, CAD tools)
  - Becoming a limiting factor in many situations, fewer unique chips can be justified

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
- 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

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
- 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

At different times, and for different applications at the same point in time, the relative balance of these factors can result in widely varying architectural choices

Course Information

All info kept up to date on the website:

`http://www.csg.csail.mit.edu/6.823`

Contact Times

- Lectures Mondays and Wednesdays
  - 1:00pm to 2:30pm in room 34-304
- Tutorial on Fridays
  - 1:00pm to 2:00pm in room 34-304
  - Attendance is optional
  - Additional tutorials will be held in evenings before quizzes
- *Quizzes on Friday (except last quiz)*
  - 1:00pm to 2:30pm in room 34-304
  - Attendance is NOT optional
- Instructor office hours
  - After class or by email appointment
- TA office hours
  - Regular weekly schedule, days/times TBD (check web site)

The course has 5 modules

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

Module 2
- Caches
- Virtual Memory

Module 3
- Complex Pipelining and Out of Order Execution
- Branch Prediction and Speculative Execution

Module 4
- Multithreading
- VLIW, EPIC
- Vector machines, GPUs
- Reliability

Module 5
- Symmetric Multiprocessors (SMPs)
- On-die networks
- Memory Models & Synchronization
- Cache Coherence Protocols

Textbook and Readings

- "Computer Architecture: A Quantitative Approach", Hennessy & Patterson, 5th ed.
  - Recommended, but not necessary

- Course website lists H&P reading material for each lecture, and optional readings that provide more in-depth coverage

Grading

- Grades are not assigned based on a predetermined curve
  - Most of you are capable of getting an A
- 80% of the grade is based on four closed book 1.5 hour quizzes (20% each)
  - The first three quizzes will be held during the tutorials; the last one during the last lecture (dates on web syllabus)
- 20% of the grade is based on three laboratory exercises weighted 5%, 5%, and 10%
- No final exam
- No final projects
  - Take 6.175 next term if you're interested in building some of these machines

Problem Sets & Labs

- Problem Sets
  - One problem set per module, not graded
  - Intended for private study and for tutorials to help prepare for quizzes
  - Quizzes assume you are very familiar with the content of problem sets
- Labs
  - Three graded labs
  - Based on widely-used PIN tool
  - Last problem open-ended challenge

Self evaluation take-home quiz

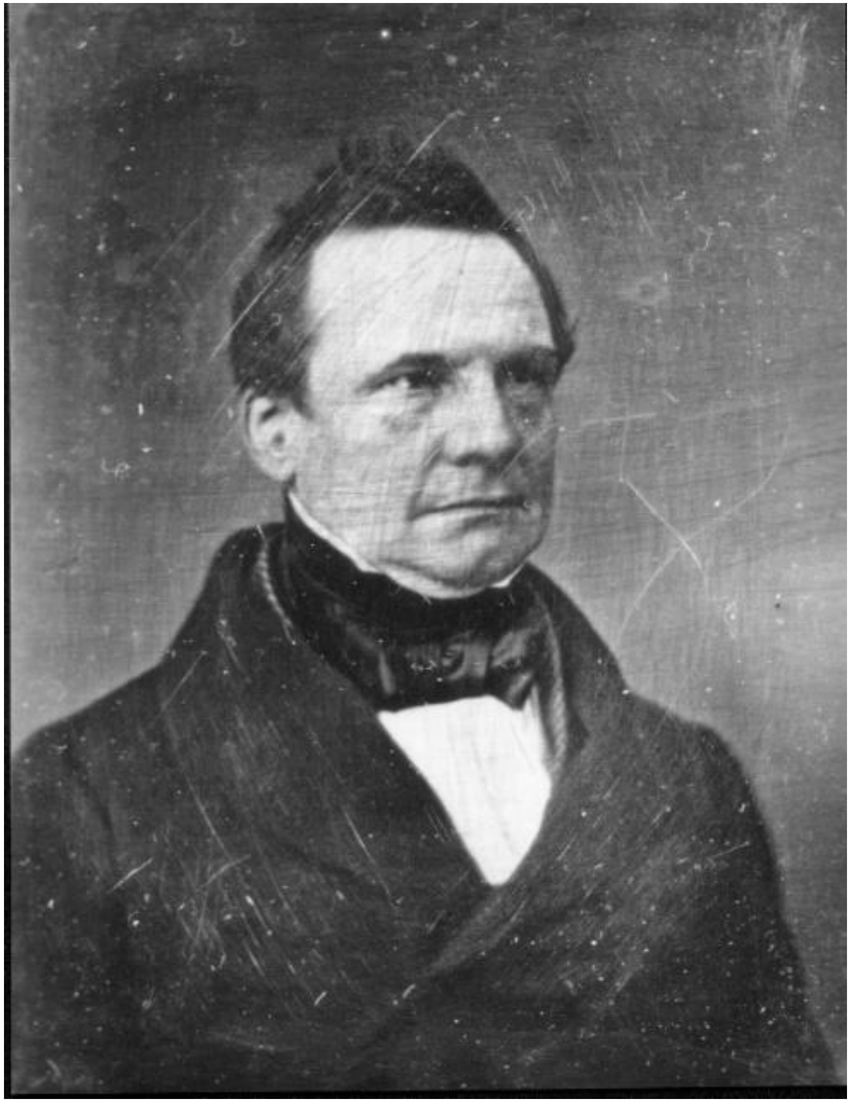
- Goal is to help you judge for yourself whether you have prerequisites for this class, and to help refresh your memory
- We assume that you understand digital logic, a simple 5-stage pipeline, and simple caches
- Please work by yourself on this quiz – not in groups
- Remember to complete self-evaluation section at end of the quiz
- Due at start of class next Monday

Please email the instructor if you have concerns about your ability to take the class

Prehistory:
Charles Babbage & Ada Byron

Charles Babbage 1791-1871

Lucasian Professor of Mathematics, Cambridge University, 1827-1839



Charles Babbage

- *Difference Engine* 1823
- *Analytic Engine* 1833
  - The forerunner of modern digital computer!

Application

- Mathematical Tables – Astronomy
- Nautical Tables – Navy

Technology

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

Architectural basis

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

-- *Weierstrass*

Difference Engine

A polynomial can be computed from difference tables

Example:

$f(n) = n^2+n+41$
$d1(n) = f(n) - f(n-1) = 2n$
$d2(n) = d1(n) - d1(n-1) = 2$

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

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

Difference Engine

A polynomial can be computed from difference tables

Example:

$f(n) = n^2+n+41$
$d1(n) = f(n) - f(n-1) = 2n$
$d2(n) = d1(n) - d1(n-1) = 2$

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

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

Difference Engine

A polynomial can be computed from difference tables

Example:

$$f(n) = n^2+n+41$$
$$d1(n) = f(n) - f(n-1) = 2n$$
$$d2(n) = d1(n) - d1(n-1) = 2$$

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

n	0	1	2	3	4 ...
d2(n)			2	2	
d1(n)		2	4	6	
f(n)	41	43	47		

Difference Engine
A polynomial can be computed from difference tables

Example:

$f(n) = n^2+n+41$
$d1(n) = f(n) - f(n-1) = 2n$
$d2(n) = d1(n) - d1(n-1) = 2$

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

n	0	1	2	3	4 ...
d2(n)			2 →	2	
d1(n)		2 →	4 →	6	
f(n)	41 →	43 →	47 →	53	

Difference Engine

A polynomial can be computed from difference tables

Example:

$f(n) = n^2+n+41$
$d1(n) = f(n) - f(n-1) = 2n$
$d2(n) = d1(n) - d1(n-1) = 2$

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

n	0	1	2	3	4 …
d2(n)			2	2	2
d1(n)		2	4	6	
f(n)	41	43	47	53	

Difference Engine

A polynomial can be computed from difference tables

Example:

$f(n) = n^2+n+41$
$d1(n) = f(n) - f(n-1) = 2n$
$d2(n) = d1(n) - d1(n-1) = 2$

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

n	0	1	2	3	4 ...
d2(n)			2	2	2
d1(n)		2	4	6	8
f(n)	41	43	47	53	

Difference Engine

A polynomial can be computed from difference tables

Example:

$$f(n) = n^2+n+41$$
$$d1(n) = f(n) - f(n-1) = 2n$$
$$d2(n) = d1(n) - d1(n-1) = 2$$

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

n	0	1	2	3	4 …
d2(n)			2	2	2
d1(n)		2	4	6	8
f(n)	41	43	47	53	61

Difference Engine
A polynomial can be computed from difference tables

Example:

$$\begin{aligned} f(n) &= n^2+n+41 \\ d1(n) &= f(n) - f(n-1) = 2n \\ d2(n) &= d1(n) - d1(n-1) = 2 \end{aligned}$$

$$\begin{aligned} d2(n) &= 2 \\ d1(n) &= d1(n-1) + d2(n) \\ f(n) &= f(n-1) + d1(n) \end{aligned}$$

n	0	1	2	3	4 ...
d2(n)			2	2	2
d1(n)		2	4	6	8
f(n)	41	43	47	53	61

What computation unit needed?

Difference Engine

A polynomial can be computed from difference tables

Example:

$$f(n) = n^2+n+41$$
$$d1(n) = f(n) - f(n-1) = 2n$$
$$d2(n) = d1(n) - d1(n-1) = 2$$

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

n	0	1	2	3	4 ...
d2(n)			2	2	2
d1(n)		2	4	6	8
f(n)	41	43	47	53	61

What computation unit needed? *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 Fair
- Can compute any 6th degree polynomial
- *Speed:* 33 to 44 32-digit numbers per minute!

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 Fair
- Can compute any 6th degree polynomial
- *Speed:* 33 to 44 32-digit numbers per minute!

- *Scheutz machine is at the Smithsonian.*
- *Diference Engine #2 – London science museum*
  - *copy at computer museum in CA*

Analytic Engine

1833: Babbage's paper was published

- *conceived during a hiatus in the development of the difference engine*

Inspiration: *Jacquard Looms*

- looms were controlled by punched cards
  - The set of cards with fixed punched holes dictated the pattern of weave ⇒ *program*
  - The same set of cards could be used with different colored threads ⇒ *numbers*

1871: Babbage dies

- The machine remains unrealized.

Analytic Engine

1833: Babbage's paper was published

- *conceived during a hiatus in the development of the difference engine*

Inspiration: *Jacquard Looms*

- looms were controlled by punched cards
  - The set of cards with fixed punched holes dictated the pattern of weave ⇒ *program*
  - The same set of cards could be used with different colored threads ⇒ *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

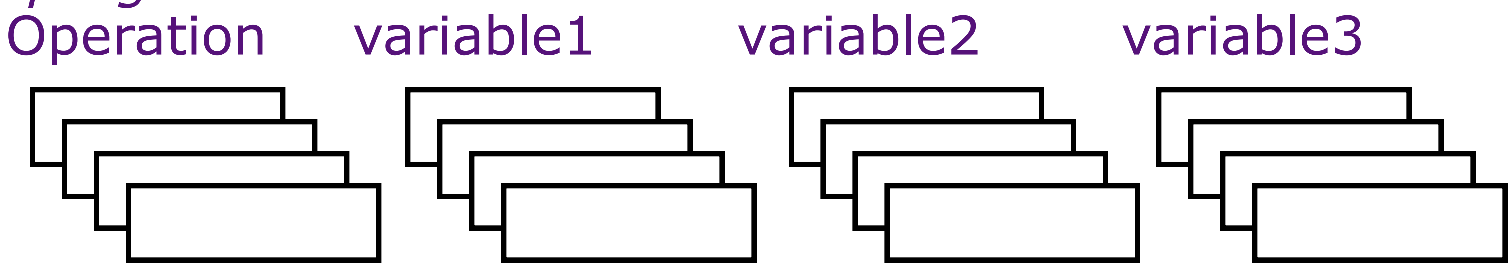
1. The *store* in which all variables to be operated upon, as well as all those quantities which have arisen from the results of the operations are placed.
2. The *mill* into which the quantities about to be operated upon are always brought.

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*



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" 1815-52

Ada's tutor was Babbage himself!

Babbage's Influence

Babbage's Influence

- Babbage's ideas had great influence later primarily because of

Babbage's Influence

- Babbage's ideas had great influence later primarily because of
  - *Luigi Menabrea,* who published notes of Babbage's lectures in Italy

Babbage's Influence

- Babbage's ideas had great 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*...."

Babbage's Influence

- Babbage's ideas had great 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*...."

- In the early twentieth century, the focus shifted to analog computers…

Babbage's Influence

- Babbage's ideas had great 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*...."

- 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*

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

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

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!

Early Developments: From *Eniac* to *IBM 701*

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

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.

Electronic Numerical Integrator and Computer (ENIAC)

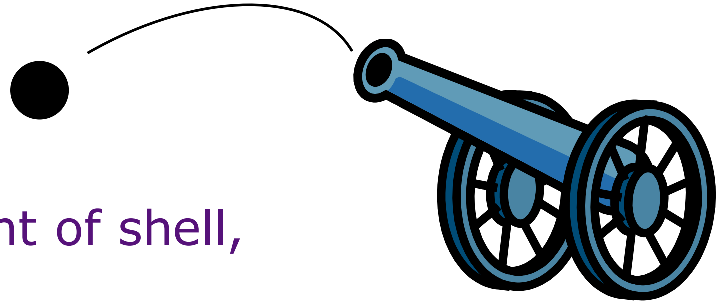
- Designed and built by Eckert and Mauchly at the University of Pennsylvania during 1943-45
- The first, completely electronic, operational, general-purpose analytical calculator!
  - 30 tons, 72 square meters, 200KW
- Performance
  - Read in 120 cards per minute
  - Addition took 200 μs, Division 6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

Electronic Numerical Integrator and Computer (ENIAC)

- 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 μs, Division 6 ms
  - 1000 times faster than Mark I
- Not very reliable!

WW-2 Effort

Application: Ballistic calculations



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

Electronic Discrete Variable Automatic Computer (EDVAC)

- 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”

Electronic Discrete Variable Automatic Computer (EDVAC)

- 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*

    $\Rightarrow$ *"program can be manipulated as data"*

Electronic Discrete Variable Automatic Computer (EDVAC)

- 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*

    $\Rightarrow$ *"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

Stored Program Computer

Program = A sequence of instructions

Stored Program Computer

Program = A sequence of instructions

How to control instruction sequencing?

Stored Program Computer

Program = A sequence of instructions

How to control instruction sequencing?

manual control calculators

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

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

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

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		

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

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 often debated by historians.

Dominant Technology Issue: *Reliability*

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

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

Dominant Technology Issue: *Reliability*

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

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

BINAC (1949)

BINAC (1949)

Two processors that checked each other for reliability.

BINAC (1949)

Two processors that checked each other for reliability.

Didn't work well because processors never agreed

BINAC (1949)

Two processors that checked each other for reliability.

Didn't work well because processors never agreed

Can two machines that don't agree each cycle both be correct?

BINAC (1949)

Two processors that checked each other for reliability.

Didn't work well because processors never agreed

Can two machines that don't agree each cycle both be correct?

Yes!

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!

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.

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 computers?

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 computers?

IBM was 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

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

Next lecture: IBM 360 & ISAs!