COMP 362 COMPUTER ARCHITECTURE

Neil Klingensmith

neil@cs.luc.edu

https://neilklingensmith.com/teaching/loyola/cs362-f2025/

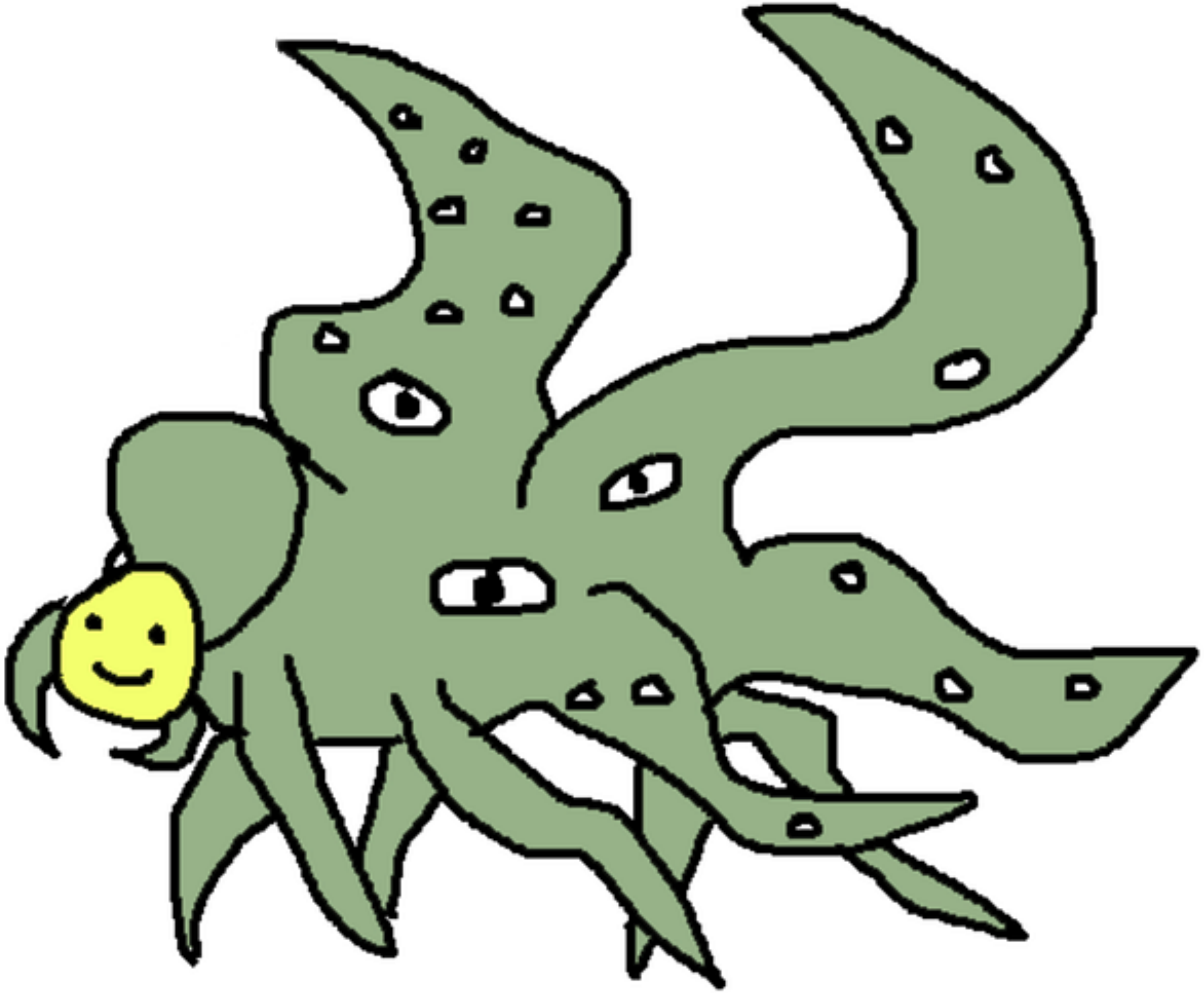
LOYOLA
UNIVERSITY CHICAGO

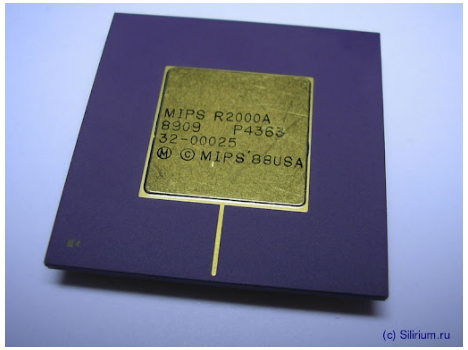
CLASS TIMING

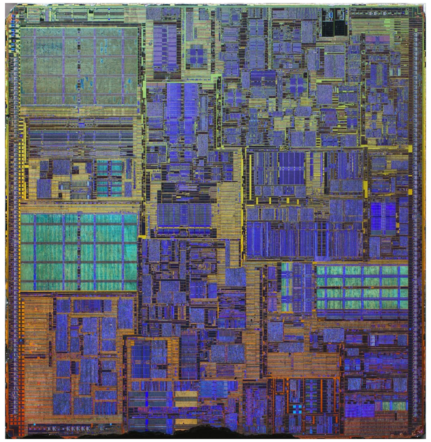
- M/W 2:30-4PM?
- NO class Friday
- Lab Wednesday 6-8 PM in Doyle Makerspace

WHAT YOU’RE GONNA LEARN

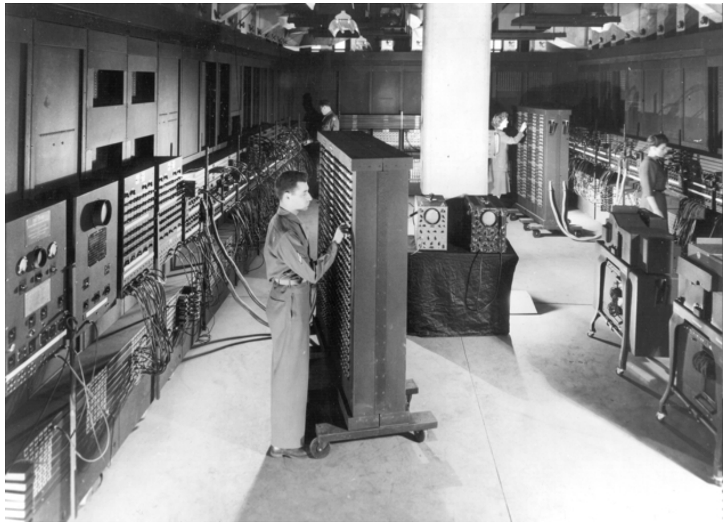
- How the CPU works
- Design issues and tradeoffs
- Verilog
- FPGA Synthesis

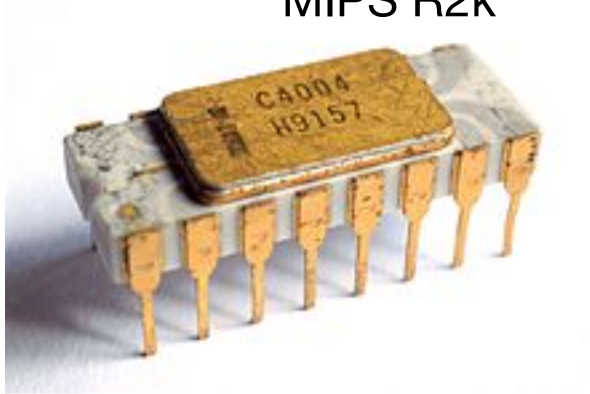


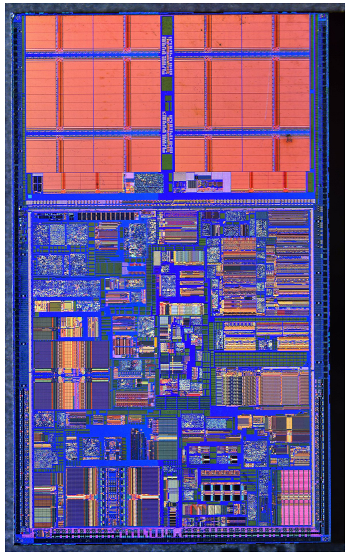




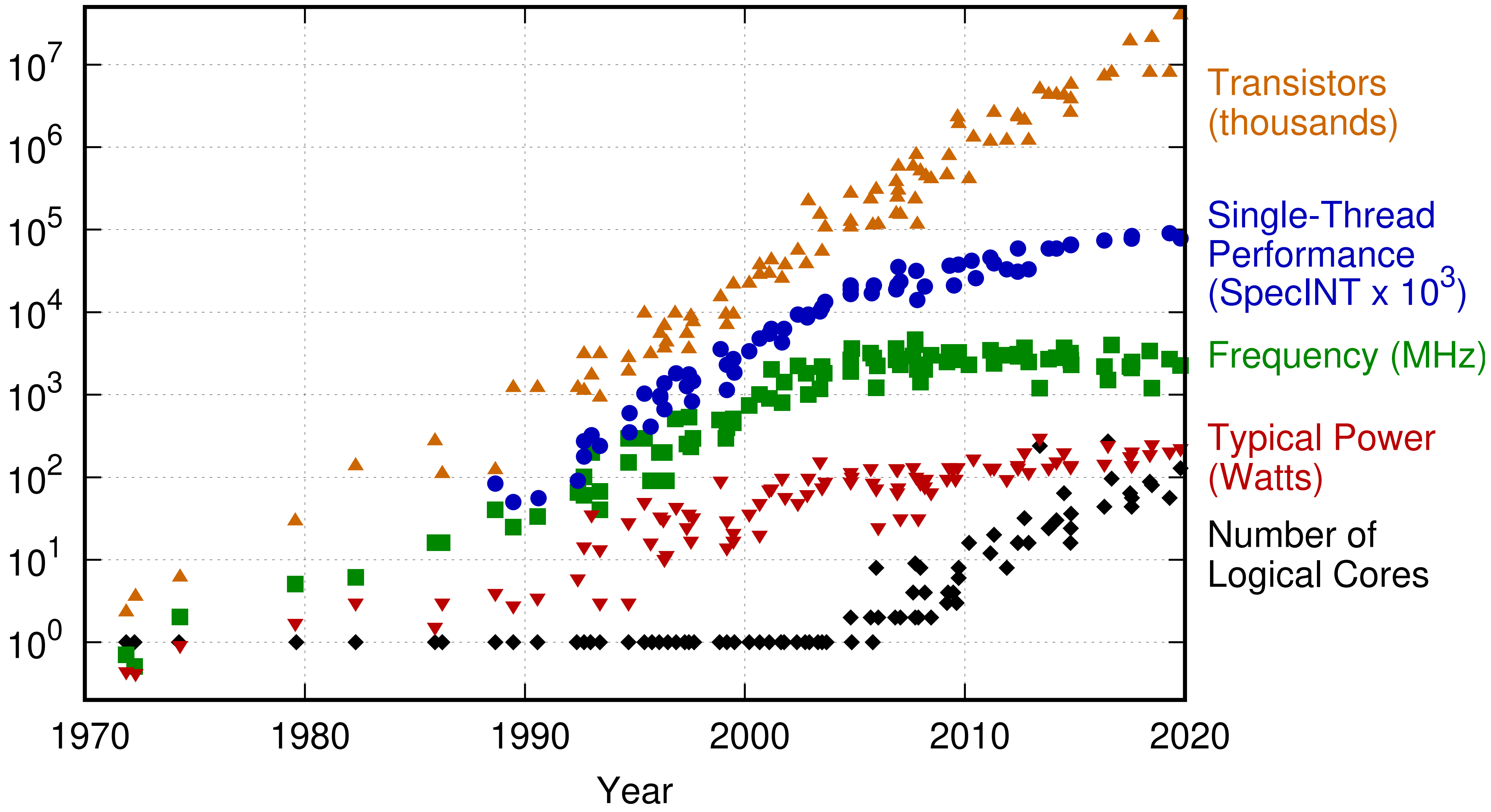
1945	1958	1964	1971	1986	1997	2000	2007	2022
ENIAC	FORTRAN	IBM 360	Intel 4004	MIPS R2k	Pentium II	Pentium 4	CUDA	RTX 4090







48 Years of Microprocessor Trend Data

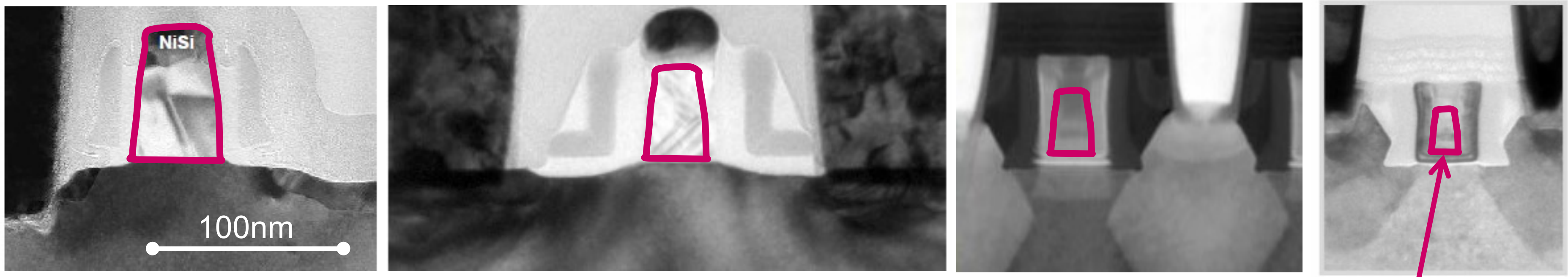


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2019 by K. Rupp

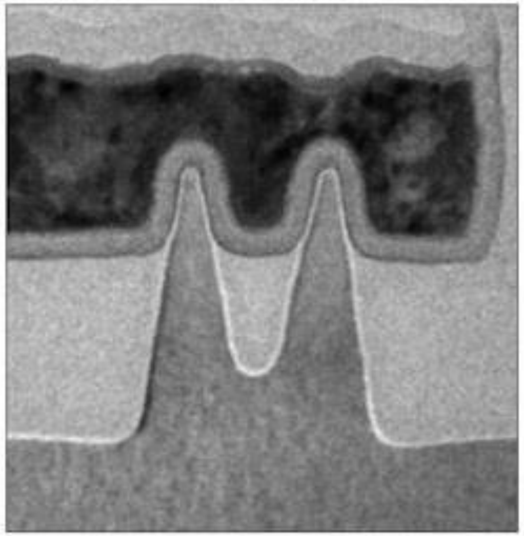
DENARD SCALING: SPEEDUP DRIVEN BY CHEMISTRY

All TEM images here have the same scale

2003	2005	2007	2009	2012
90nm node	**65nm node**	**45nm node**	**32nm node**	**22nm node** (FINfet)



0.7x scaling



22 nm 1st Generation Tri-gate Transistor

- Very little change in physical gate length, only ~0.9x per node
- The gate pitch is scaling fast, as 0.7x per node and area scales as 0.5x
- Most of the transistor innovation is in stress engineering and HKMG

2000

ATMEL

8-bit, 8 MHz
2-stage pipeline

intel inside pentium III

400 MHz
10-stage 2-wide OoO pipeline

2005

arm CPU CORTEX-M3

25 MHz
3-stage pipeline

intel inside pentium 4

2 GHz
HyperThreading
31-stage 3-wide OoO pipeline

2010

arm CORTEX®-M4

100 MHz
3-stage pipeline
floating point + DMA

intel Core 2 Duo inside

2 GHz
HyperThreading
18-stage 4-wide OoO pipeline

2015

arm CORTEX®-M7

300 MHz
6-stage 2-wide in-order pipeline
floating point, DMA, DRAM IF

intel inside CORE i7

2 GHz
HyperThreading
18-stage 9-wide OoO pipeline

2020

?

2.5 GHz
(no HyperThreading)
18-stage 6-wide?
OoO pipeline

8-bit vs 32-bit

2000 — 8-bit, 8 MHz
2-stage pipeline

400 MHz
10-stage 2-wide OoO pipeline

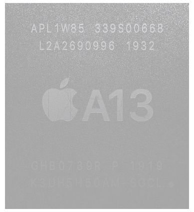
2005

2010

2015

2020

?



2000

8-bit vs 32-bit

2005

25 MHz
3-stage pipeline

2 GHz
HyperThreading
31-stage 3-wide OoO

Low Power
vs
High Power

2010

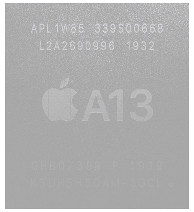
100 MHz
3-stage pipeline
floating point + DMA

2 GHz
HyperThreading
18-stage 4-wide OoO

2015

2020

?



2000 ATMEL

intel inside pentium III

8-bit vs 32-bit

2005 arm CPU CORTEX-M3

intel inside pentium 4

Low Power
vs
High Power

2010 arm CORTEX®-M4

intel Core 2 Duo inside

2015 arm CORTEX®-M7

300 MHz
6-stage 2-wide in-order pipeline
floating point, DMA, DRAM IF

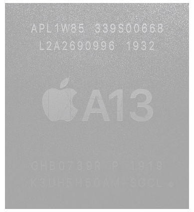
intel inside CORE i7

2 GHz
HyperThreading
18-stage 9-wide OoO

TLB vs No TLB

2020

?

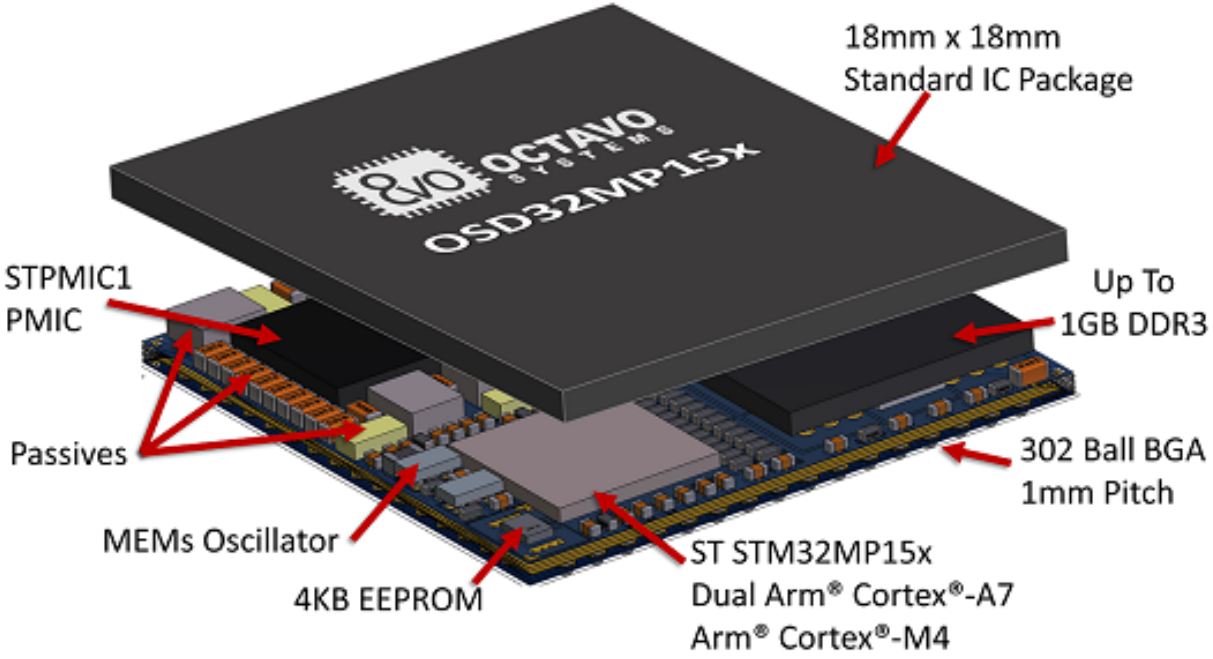


COMPUTER ARCHITECTURE STARTUPS

LUMINARY MICRO

- First company to produce ARM Cortex-M3 MCUs
- Founded in 2004 by Jim Reinhart and Jean Anne Booth
- Raised $44 MM from 4 investors
- Acquired by TI in 2009 with ~70 employees

OCTAVO SYSTEMS



- Build computer-in-package
- Founded 2015 by some old guys who defected from TI

SiFive

- Make chips based on RISC-V CPU
- Founded by some guys who quit their PhDs at Berkeley
- Raised $130MM

N
VON

- Make AI accelerator chips that don't use von Neumann architecture.
- Founded by guys from Dartmouth in 2022.

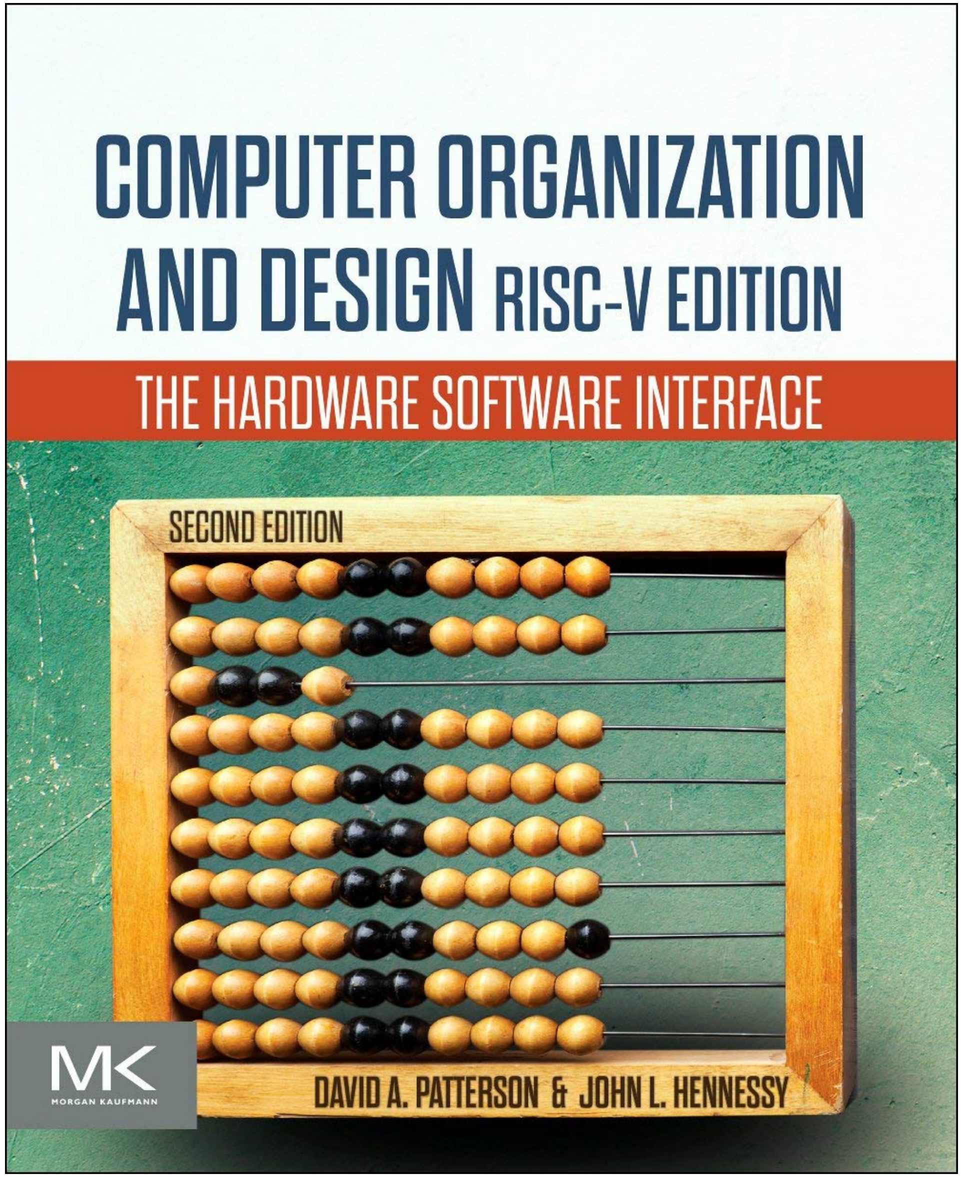
OBVIOUS OPPORTUNITIES

- Multicore microcontrollers with shared memory
- Microcontroller with low-power TLB
- Open-source GPU (a la RISC-V/SiFive)

COURSE ADMINISTRIVIA

TEXTBOOK

Get it.



LAB

- Lab located in Doyle Makerspace
- Computers in that room have ModelSim

INTEL QUARTUS LITE

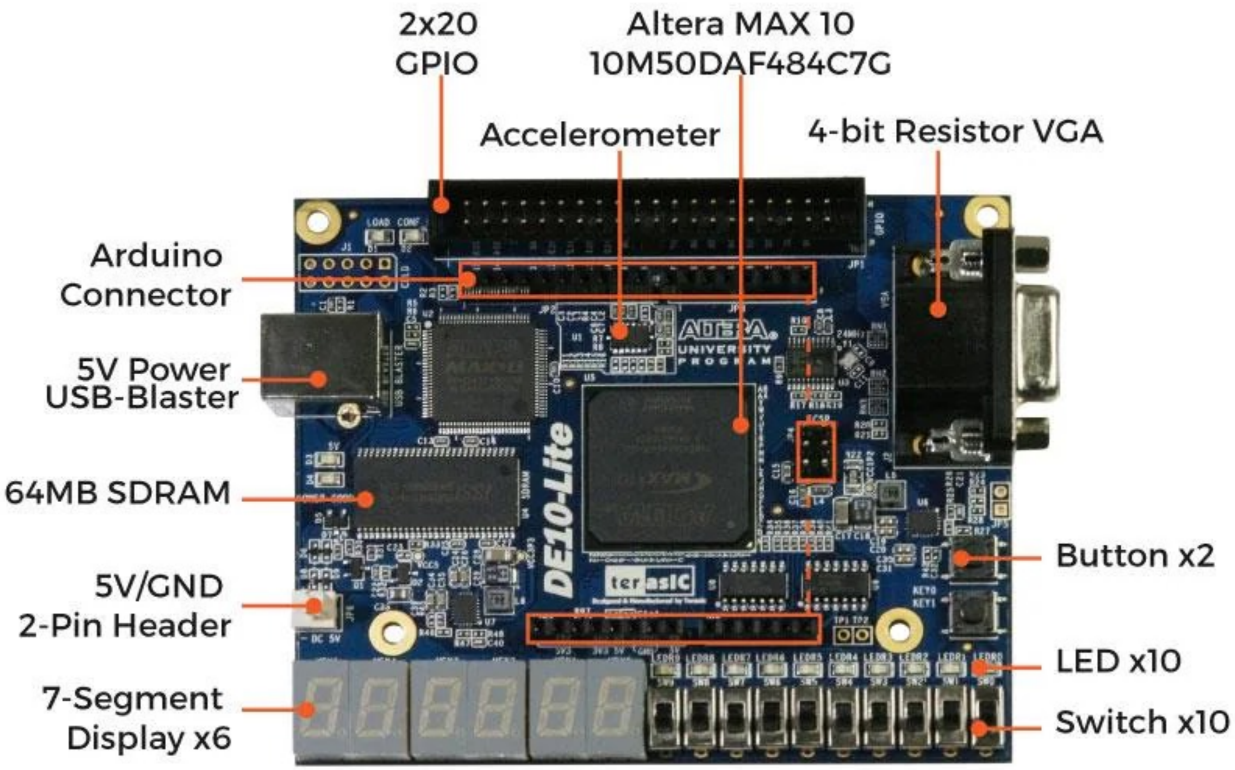
Runs in Windows & Linux

Installing it is a mission.

(Intel site was down as of this morning...)

TERASIC DE10-LITE

- 50k Logic Elements (gates)
- 1.6 Mbytes SRAM
- 5.8 Mbytes flash
- 144 hardware multipliers
- 4 PLLs



GRADING

- No quizzes or exams. Your whole grade is based on homework and final project.
- No partial credit for code that doesn't compile.
- Start homework on Tuesday/Wednesday so you can get help on Thursday in lab if you get stuck.
- Slop Days: Everyone gets 5 slop days. Each slop day allows you to turn in an assignment 24 hours late.

Category	Weight
Homework	40%
Participation	20%
Course Project	40%

DEMO DAYS

- Course project will be done in 3 segments.
- For each segment we will do a demo day.

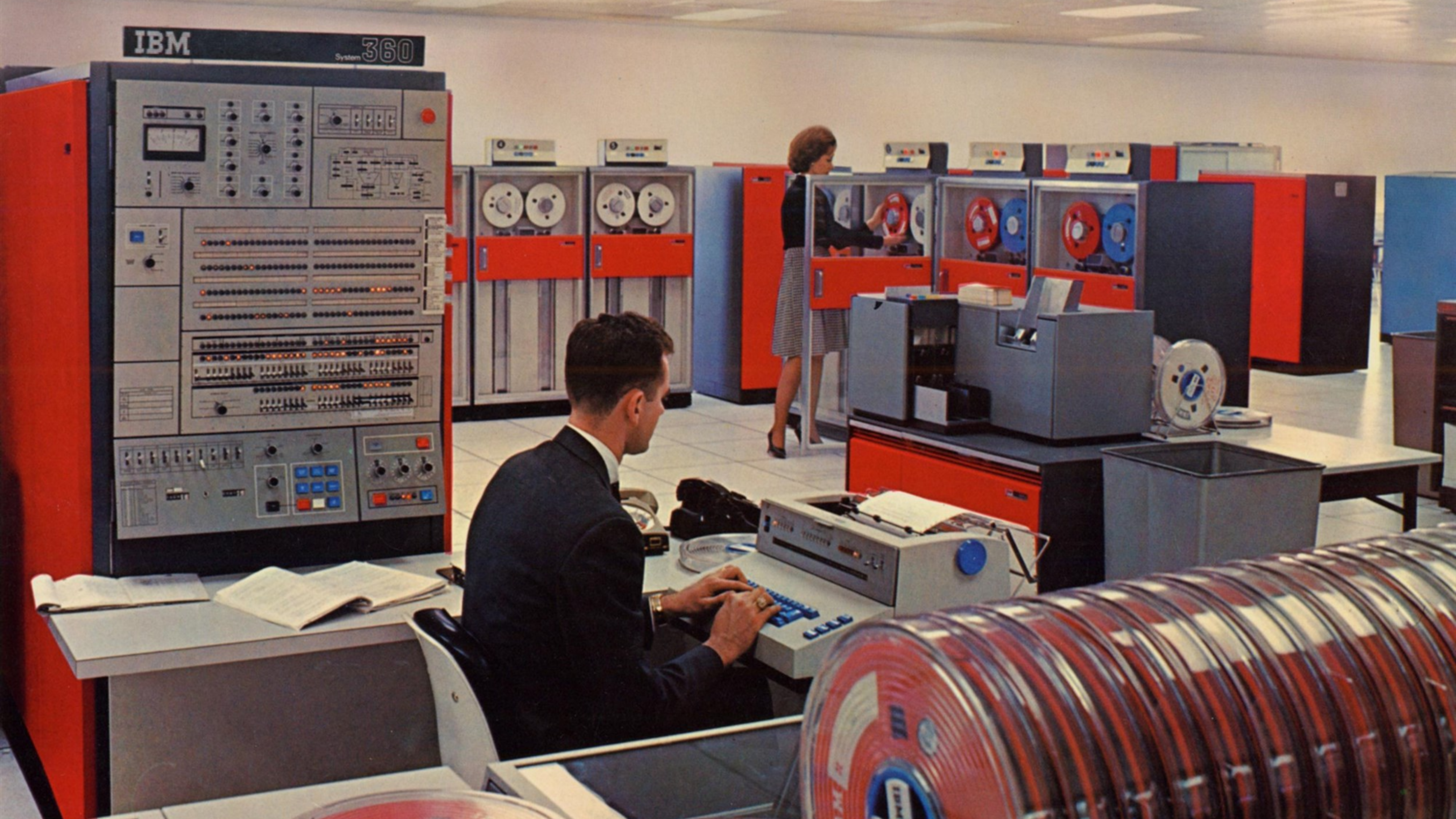
ABOUT RISC-V

RISC-V

- Open-source RISC processor
  - Unlike ARM, you can design your own RISC-V compatible CPU without paying anyone
  - Instruction Set Architecture fixes some ugly bugs that have been around in other CPUs like OpenRISC and MIPS.
- `gcc` port is available (you should install it)
- Documentation is available

INSTRUCTION SET ARCHITECTURES


IBM 7090 DATA PROCESSING SYSTEM


IBM
System 360

INSTRUCTION SET ARCHITECTURE (ISA)

- Unified (among CPU models) & well-defined interface btw software and hardware
  - register names, instruction mneumonics, memory model, etc.
- If hardware changes, old software will still work
- Example: x86 programs from 1985 still run on brand new Core i7

MICROARCHITECTURE (THIS CLASS)

- Block diagram of CPU
- Underlying hardware that implements the ISA

THEMES

3 BIG IDEAS IN ARCHITECTURE

- Pipelining
- Parallelism
- Caching

1. OVERHEAD KILLS

- 20% of instructions are branches
- 2/5 pipeline stages (fetch and decode) are overhead

2. LOCALITY KILLS

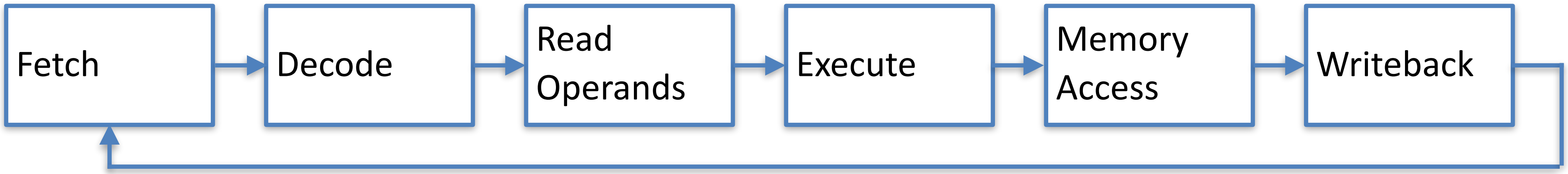
- Instruction dependence chains limit parallelism

INSTRUCTIONS

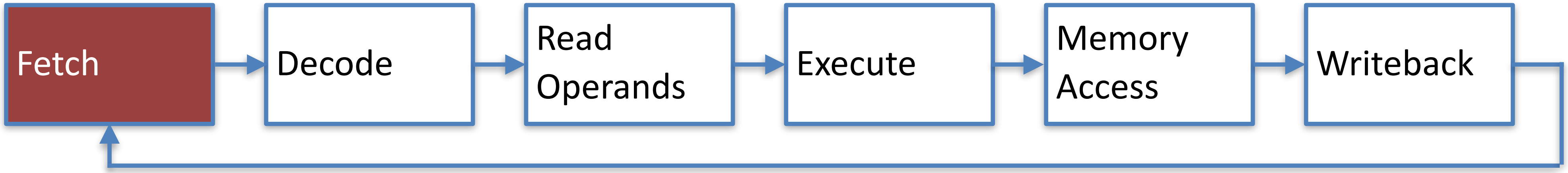
KINDS OF INSTRUCTIONS

- Arithmetic
  - Add, subtract, multiply, divide
- Logic
  - AND, OR, NOT, XOR
- Shifts
  - Left shift, right shift, rotate, etc.
- Control
  - Branch/Jump
  - Procedure calls
- Memory Accesses
  - Load/store

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.



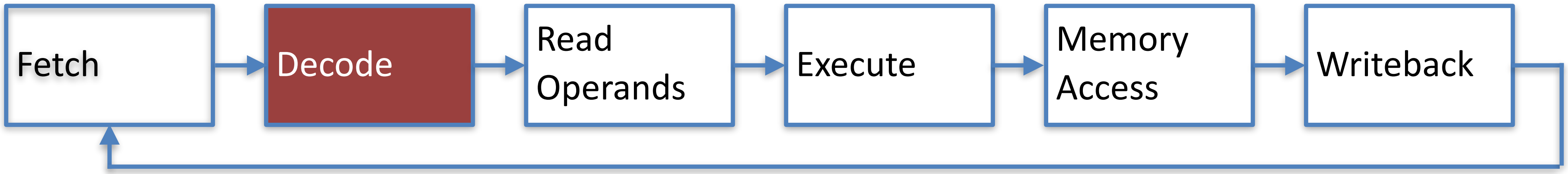
THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.



0x8900

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.

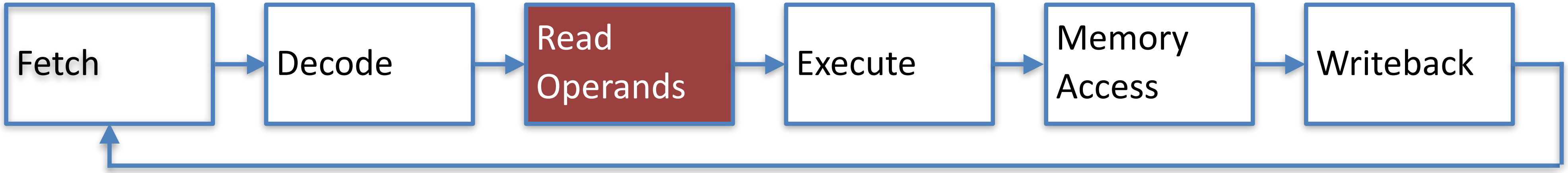
```
mov [si+bx],ax
```




```
0x8900
```

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.

```
mov [si+bx],ax
```

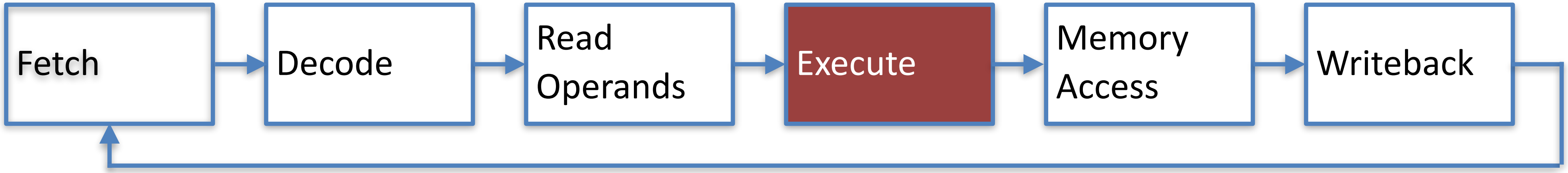



```
0x8900
```

```
ax: 0x1234
bx: 0x0002
si: 0x5678
```

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.

`mov [si+bx],ax` EA: 0x567A



0x8900

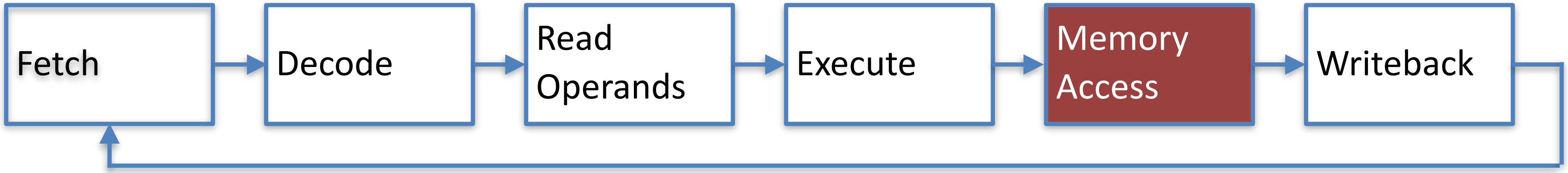
ax: 0x1234

bx: 0x0002

si: 0x5678

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.

`mov [si+bx],ax`   `EA: 0x567A`



`0x8900`

`ax: 0x1234`

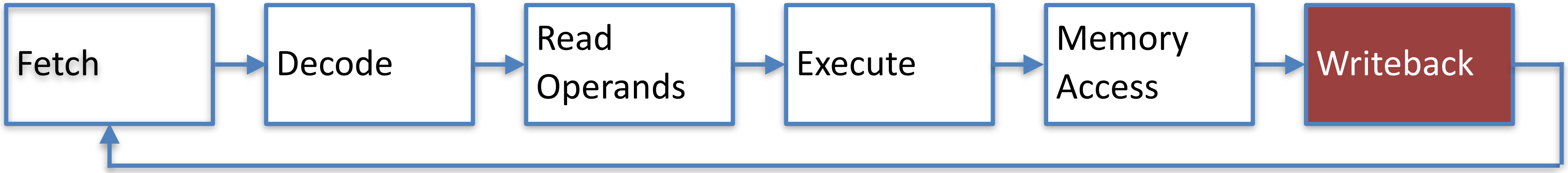
`bx: 0x0002`

`si: 0x5678`

`mem[0x567A] <- 0x1234`

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.

mov [si+bx],ax   EA: 0x567A



0x8900

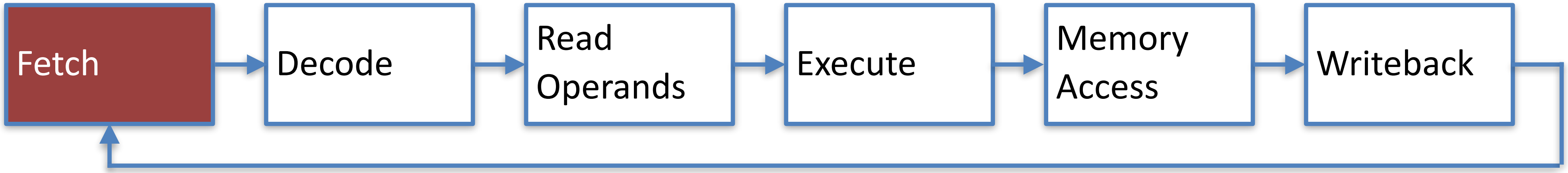
ax: 0x1234

bx: 0x0002

si: 0x5678

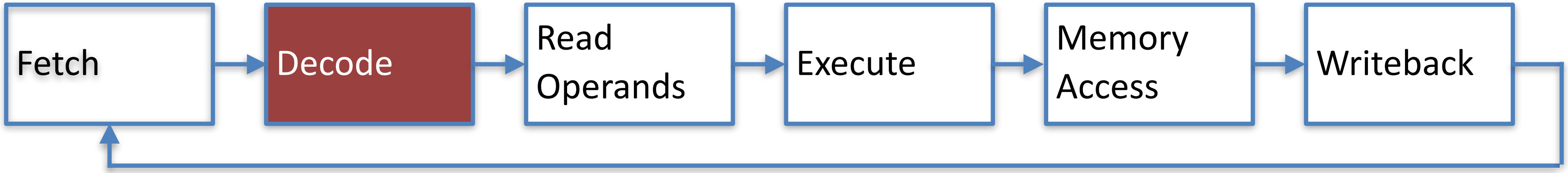
mem[0x567A] <- 0x1234

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.



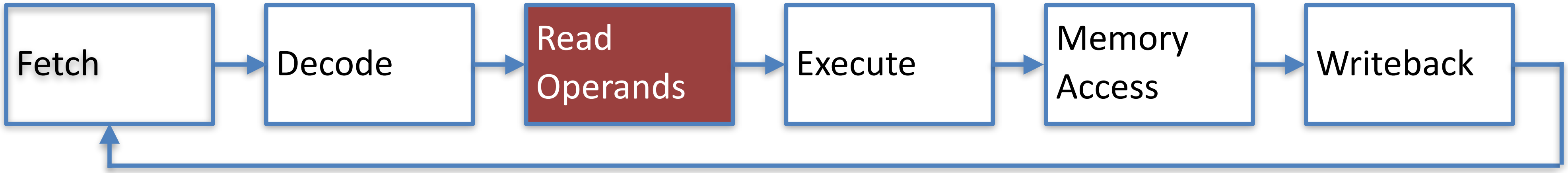
`0x89e5`

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.



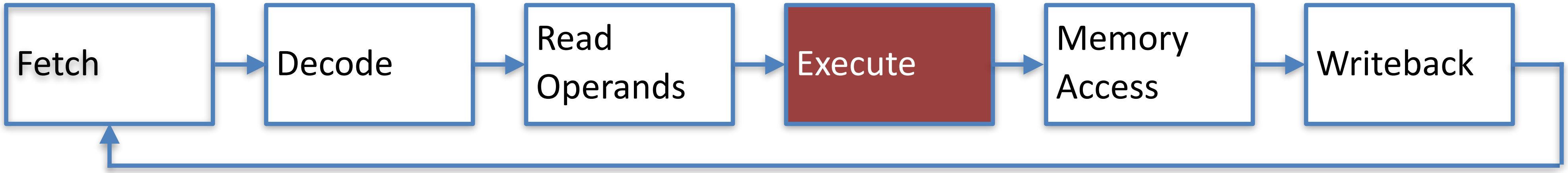

```
0x89e5   mov bp,sp
```

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.



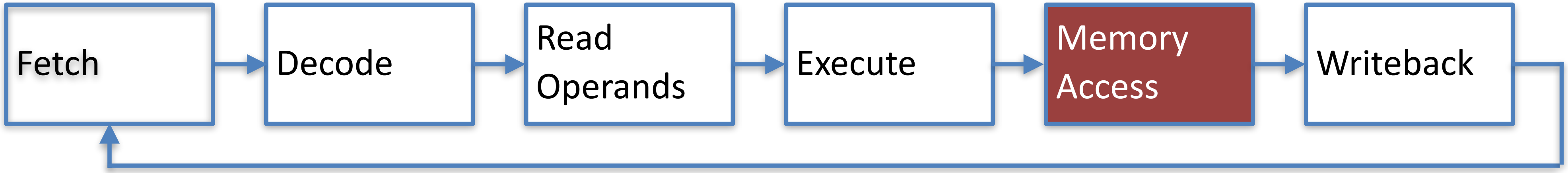
`0x89e5`   `mov bp,sp`   `sp contains 0x1000`

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.



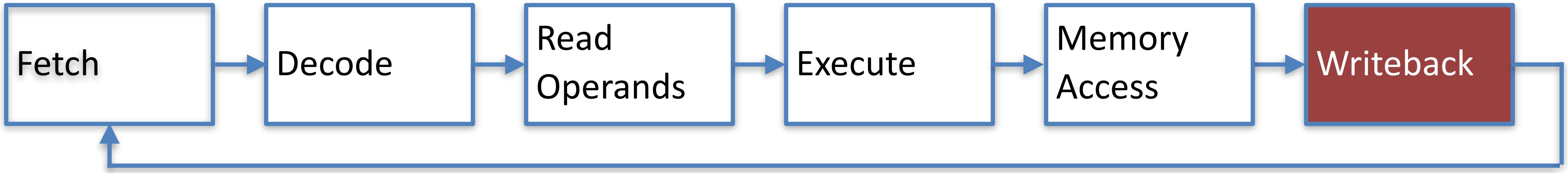
`0x89e5` `mov bp,sp` `sp contains 0x1000`

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.



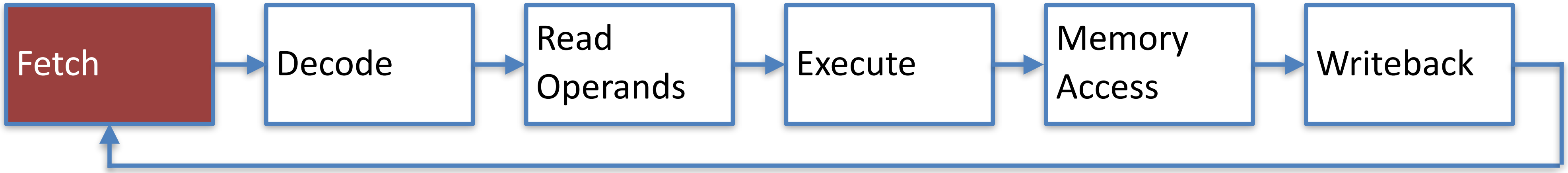

```
0x89e5    mov bp,sp    sp
                       contains
                       0x1000
```

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.



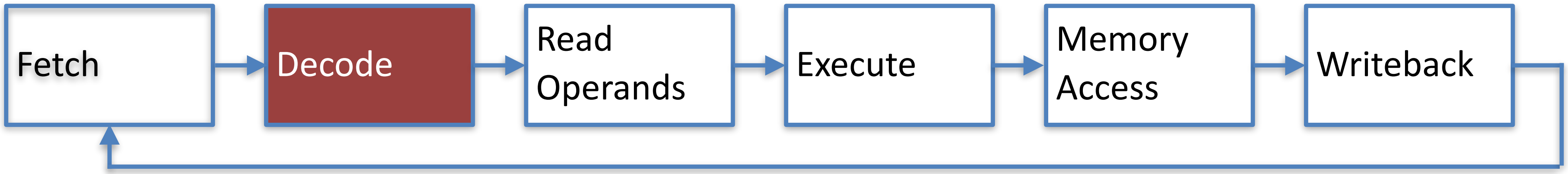
`0x89e5`   `mov bp,sp`   `sp contains 0x1000`   `Set BP<-0x1000`

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.



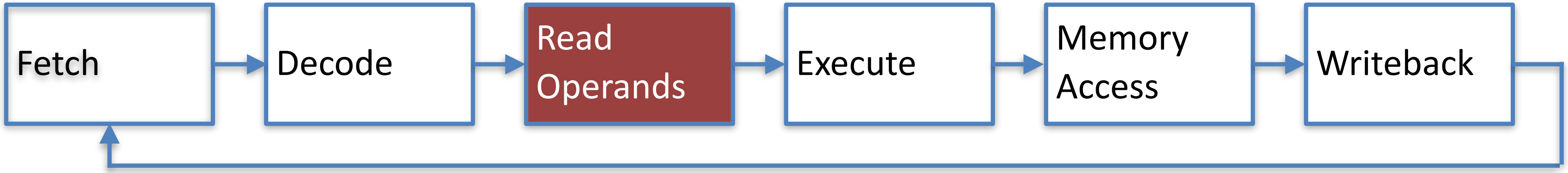
`0x30ff`

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.



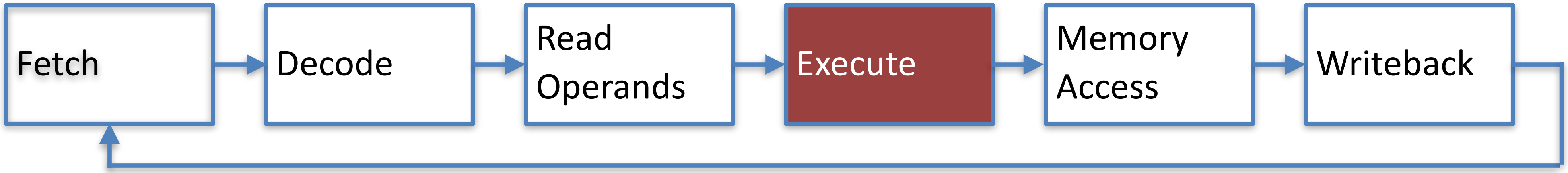

```
0x30ff   xor bh,bh
```

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.



0x30ff xor bh,bh bh contains 0x7199

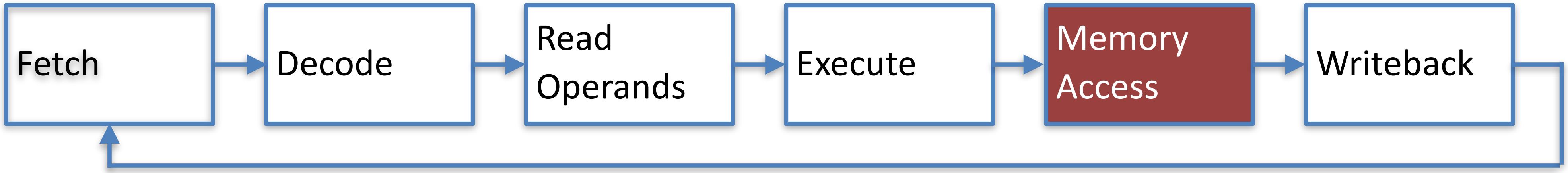
THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.



Fetch	Decode	Read Operands	Execute

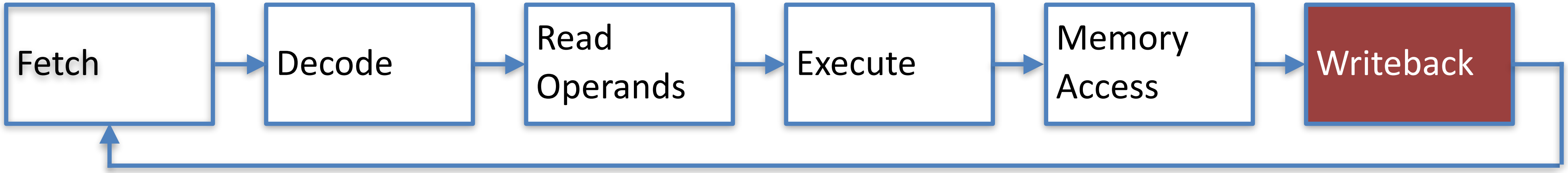
`0x30ff`	`xor bh,bh`	`bh contains 0x7199`	`0x7199 XOR 0x7199`
			 `________`
			 `0x0000`

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.




```
0x30ff    xor bh,bh    bh          0x7199
                       contains    XOR
                       0x7199      0x7199
                                   ______
                                   0x0000
```

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.



Fetch	Decode	Read Operands	Execute	Memory Access	Writeback
0x30ff	xor bh,bh	bh contains 0x7199	0x7199 XOR 0x7199 ________ 0x0000		Set BH<-0x0000