

CS 3410: Computer System Organization and Programming

Hakim Weatherspoon

Spring 2011

Computer Science

Cornell University

Information

- Instructor: Hakim Weatherspoon
(hweather@cs.cornell.edu)
- Tu/Th 1:25-2:40
- Phillips 101

Course Objective

- Bridge the gap between hardware and software
 - How a processor works
 - How a computer is organized
- Establish a foundation for building higher-level applications
 - How to understand program performance
 - How to understand where the world is going

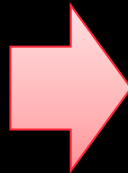
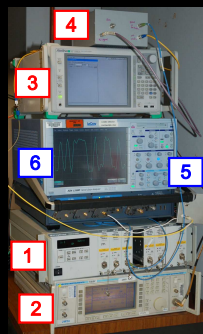
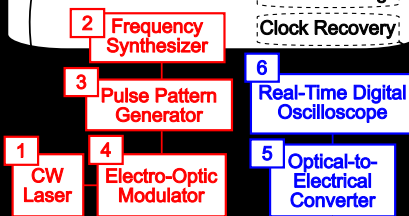
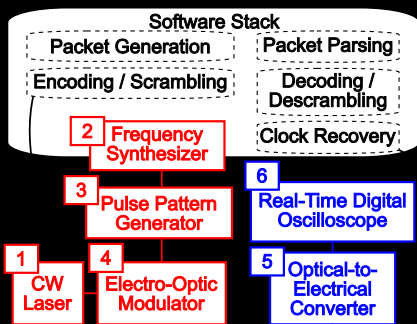
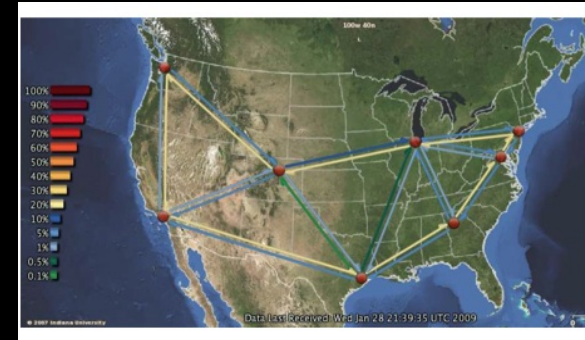
Who am I?



- Prof. Hakim Weatherspoon
 - (Hakim means Doctor, wise, or prof. in Arabic)
 - Background in Education
 - Undergraduate University of Washington
 - Played Varsity Football
 - Some teammates collectively make \$100's of millions
 - I teach!!!
 - Graduate University of California, Berkeley
 - Some class mates collectively make \$100's of millions
 - I teach!!!
 - Background in Operating Systems
 - Peer-to-Peer Storage
 - Antiquity project - Secure wide-area distributed system
 - OceanStore project – Store your data for 1000 years
 - Network overlays
 - Bamboo and Tapestry – Find your data around globe
 - Tiny OS
 - Early adopter in 1999, but ultimately chose P2P direction

Who am I?

- Cloud computing/storage
 - Optimizing a global network of data centers
 - Cornell National λ -Rail Rings testbed
 - Software Defined Network Adapter
 - Energy: KyotoFS/SMFS
- Antiquity: built a global-scale storage system

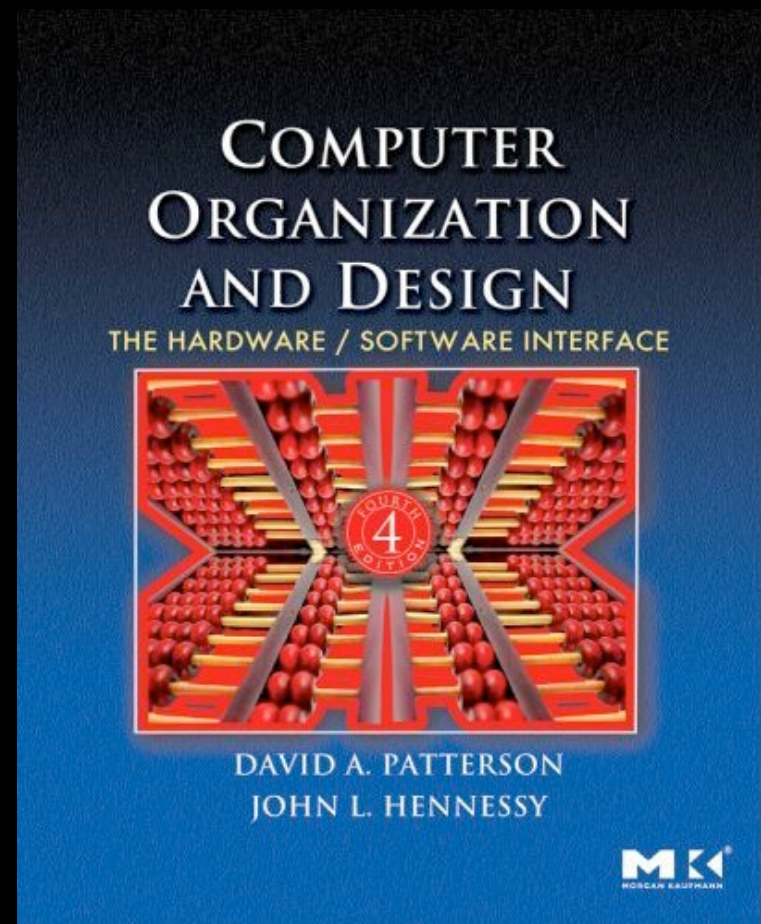


Course Staff

- cs3410-staff-l@cs.cornell.edu
 - TAs
 - Han Wang (hwang@cs.cornell.edu)
 - Bo Peng (bpeng@cs.cornell.edu)
 - Jun Erh (je96@cornell.edu)
 - Undergraduate consultants
 - Ansu Abraham (aaa98@cornell.edu)
 - Ethan Kao (ek382@cornell.edu)
 - Peter Tseng (pht24@cornell.edu)
 - Jiaqi Zhai (jz392@cornell.edu)
- Administrative Assistant:
- Angela Downing (angela@cs.cornell.edu)

Book

- Computer Organization and Design
 - The Hardware/Software Interface
- David Patterson, John Hennessy
 - Get the 4th Edition



Grading

- 4 Programming Assignments (35-45%)
 - Work in **groups** of two
- 4-5 Homeworks Assignments (20-25%)
 - Work **alone**
- 2 prelims (30-40%)
- Discretionary (5%)

Grading

- Regrade policy
 - Submit written request to lead TA, and lead TA will pick a different grader
 - Submit another written request, lead TA will regrade directly
 - Submit *yet* another written request for professor to regrade.

Administrivia

- <http://www.cs.cornell.edu/courses/cs3410/2011sp>
 - Office Hours / Consulting Hours
 - Lecture slides & schedule
 - Logisim
 - CSUG lab access (esp. second half of course)

- **Sections**

T	2:55 – 4:10pm	Hollister 372
W	3:35 – 4:50pm	Upson 215
R	11:40 – 12:55pm	Hollister 372
R	2:55 – 4:10pm	Hollister 368
F	2:55 – 4:10pm	Phillips 213
TBD		

- Will cover new material
- Next week: intro to logisim

Communication

- Email
 - `cs3410-staff-1@cs.cornell.edu`
 - The email alias goes to me and the TAs, not to whole class
- Assignments
 - CMS: `http://cms.csuglab.cornell.edu`
- Newsgroup
 - `cornell.class.cs3410`
 - For students

Sections & Projects

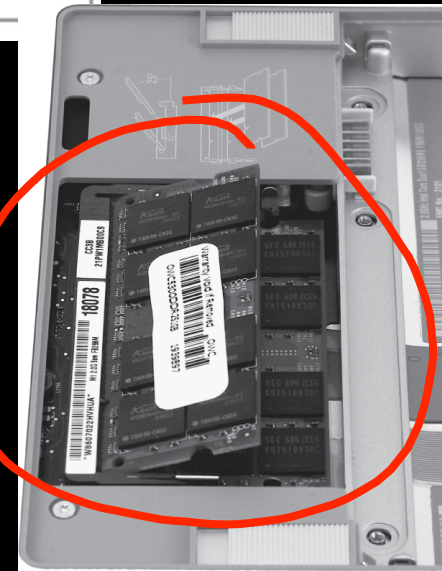
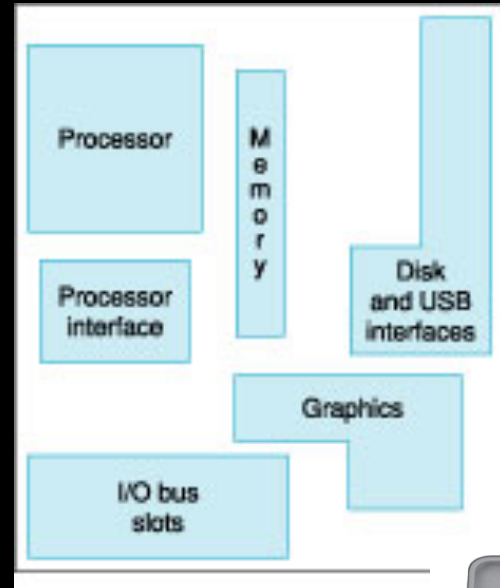
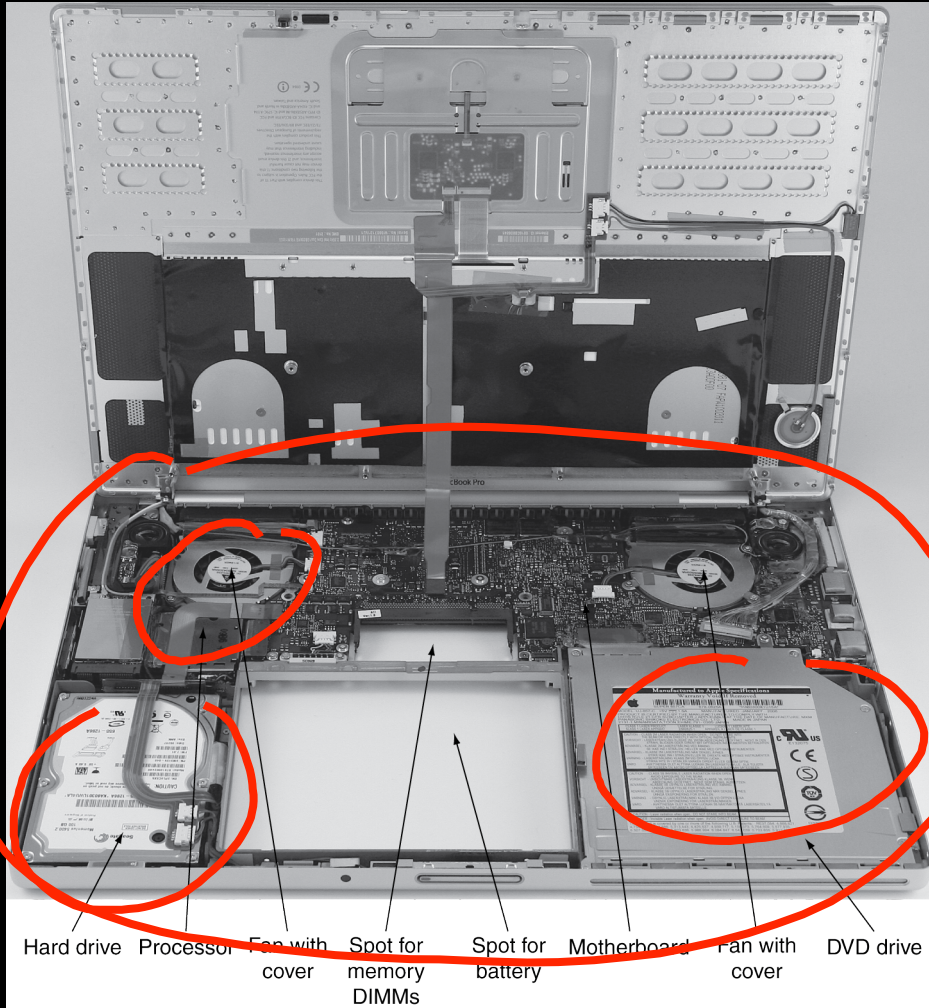
- Sections start next week
 - But can go this week to find a project partner
- Projects will be done in two-person teams
 - We will pair you up if you don't have a preferred partner
 - Start early, time management is key
 - Manage the team effort

Academic Integrity

- All submitted work must be your own
 - OK to study together, but do not share soln's
 - Cite your sources
- Project groups submit joint work
 - Same rules apply to projects at the group level
 - Cannot use of someone else's soln
- Closed-book exams, no calculators
- Stressed? Tempted? Lost?
 - Come see me before due date!

Plagiarism in any form will not be tolerated

Computer System Organization



Compilers & Assemblers

C

```
int x = 10;  
x = 2 * x + 15;
```

const
↓ zero

compiler

MIPS
assembly
language

```
addi r5, r0, 10  
mulr r5, r5, 2  
addi r5, r5, 15
```

r5 = 10 + 10
*r5 = 10 * 2*
r5 = 10 + 15

assembler

MIPS
machine
language

```
001000000000010100000000000000001010  
0000000000000001010010100001000000  
0010000010100101000000000000001111
```

op = addi, r0, r5 = 5

00101
2² + 2⁰ =

addi, r5, r5

15 = 2³ + 2² + 2¹ + 2⁰

Compilers

C

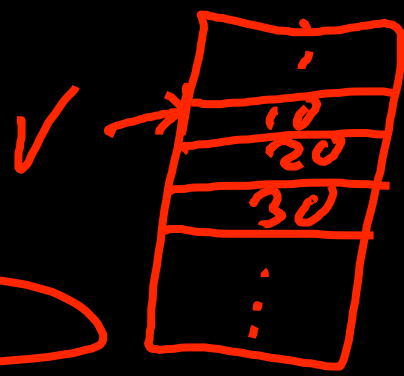


MIPS
assembly language

load word

```
int sum3(int v[]) {  
    return v[0] +  
           v[1] +  
           v[2];  
}
```

```
main() {  
    ...  
    int v[] = ...;  
    int a = sum3(v);  
    v[3] = a;  
    ...  
}
```



```
sum3:  
lw    r9, 0(r5) V[C]  
lw    r10, 4(r5) V[C]  
lw    r11, 8(r5) V[C]  
add   r3, r9, r10  
add   r3, r3, r11  
jr    r31
```

```
main:  
...  
addi  r5, r0, 1000  
jal  sum3  
sw    r3, 12(r5) V[C]  
...
```


Assemblers

MIPS
assembly language

assembler

MIPS
machine language

sum3:

```
lw    r9, 0(r5)
lw    r10, 4(r5)
lw    r11, 8(r5)
add   r3, r9, r10
add   r3, r3, r11
jr    r31
```

(lw r5 r9)

```
1000110010101010010000000000000000
1000110010101010000000000000000100
10001100101010101100000000000001000
000000010010101000001100000100000
000000000011010110001100000100000
0000001111100000000000000000001000
```

main:

```
...
addi  r5, r0, 1000
jal   sum3
sw    r3, 12(r5)
...
```

```
...
...
001000000000001010000001111101000
0000110000001000000000000000000000
1010110010100011000000000000001100
...
```

Computer System Organization

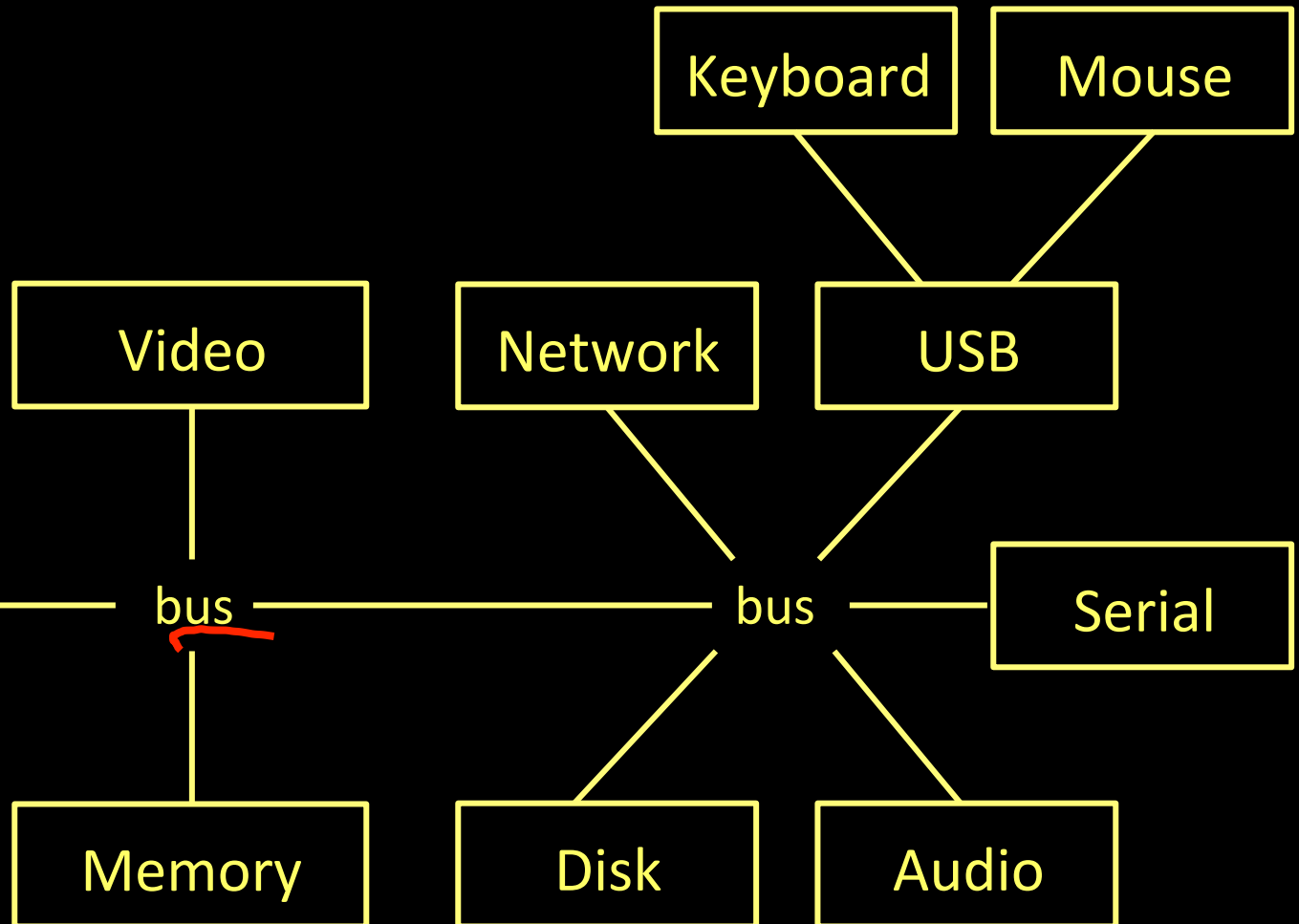
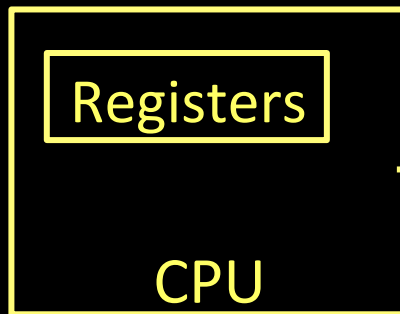
Computer System = ?

Input +

Output +

Memory +

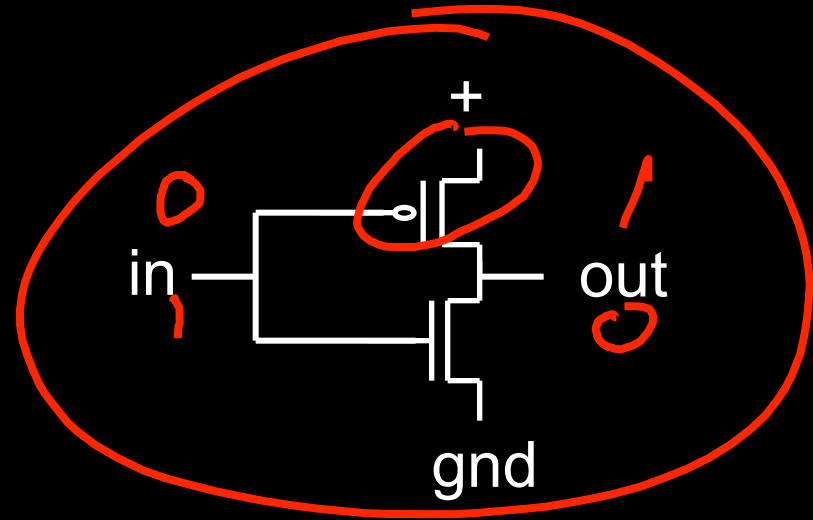
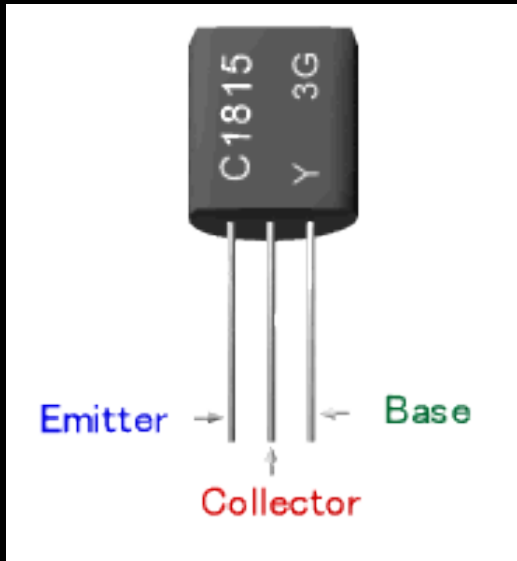
Datapath +
Control



Instruction Set Architecture

- ISA
 - abstract interface between hardware and the lowest level software
 - user portion of the instruction set plus the operating system interfaces used by application programmers

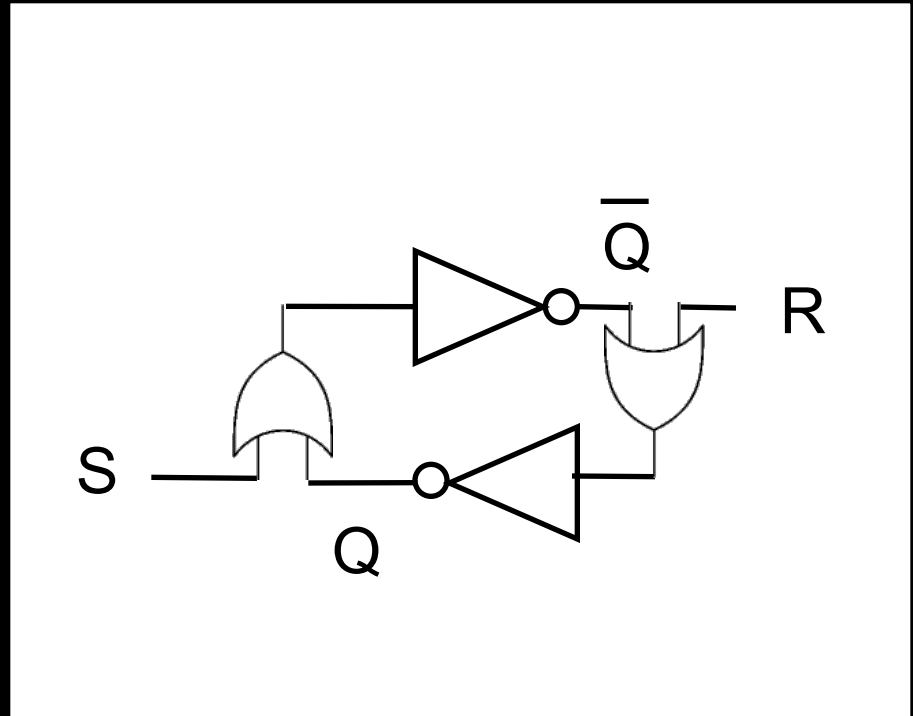
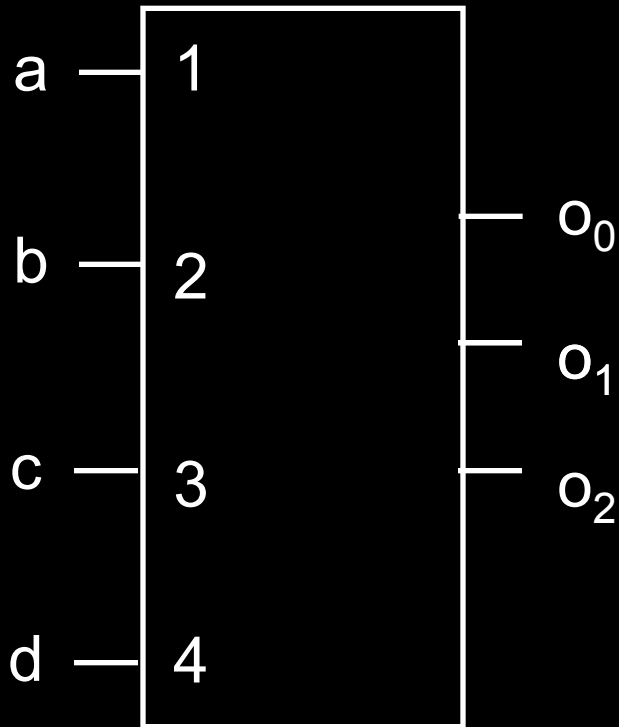
Transistors and Gates



In	Out
0	1
1	0

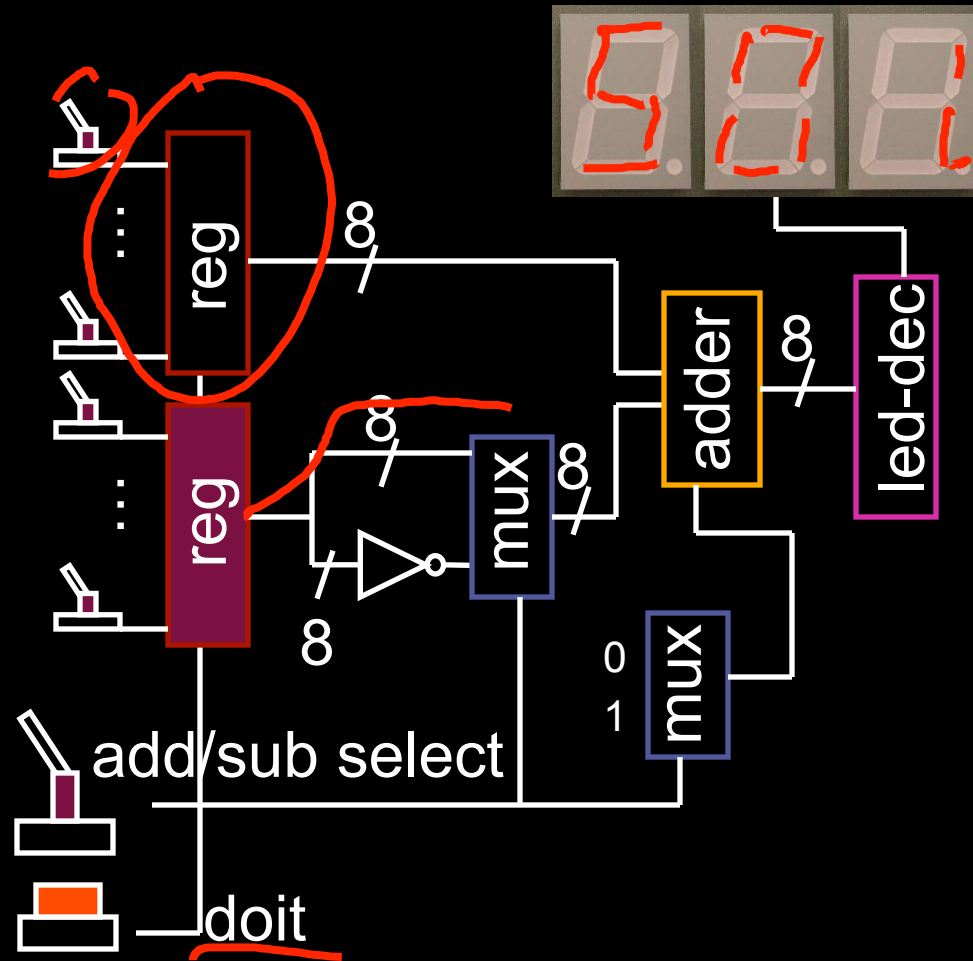
Truth table

Logic and State



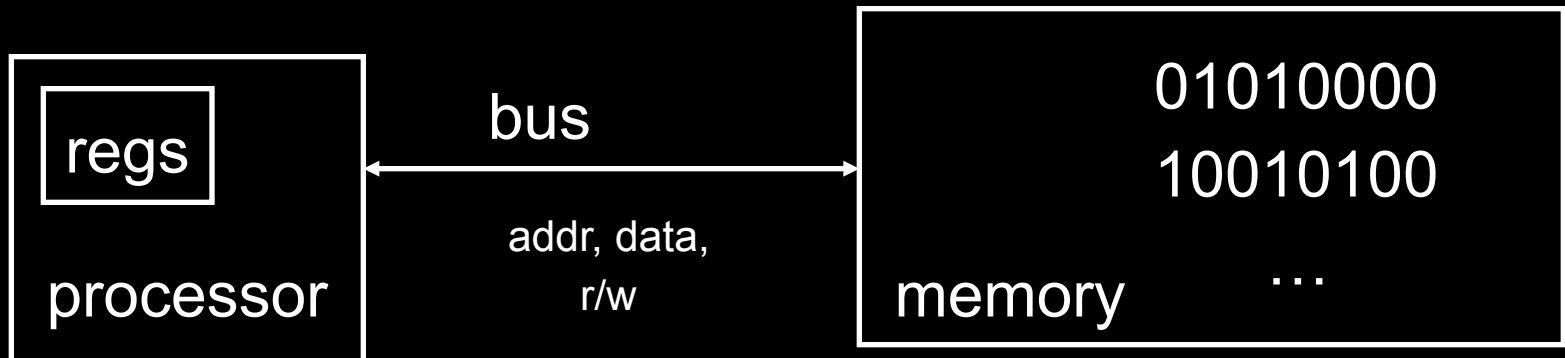
Latch

A Calculator

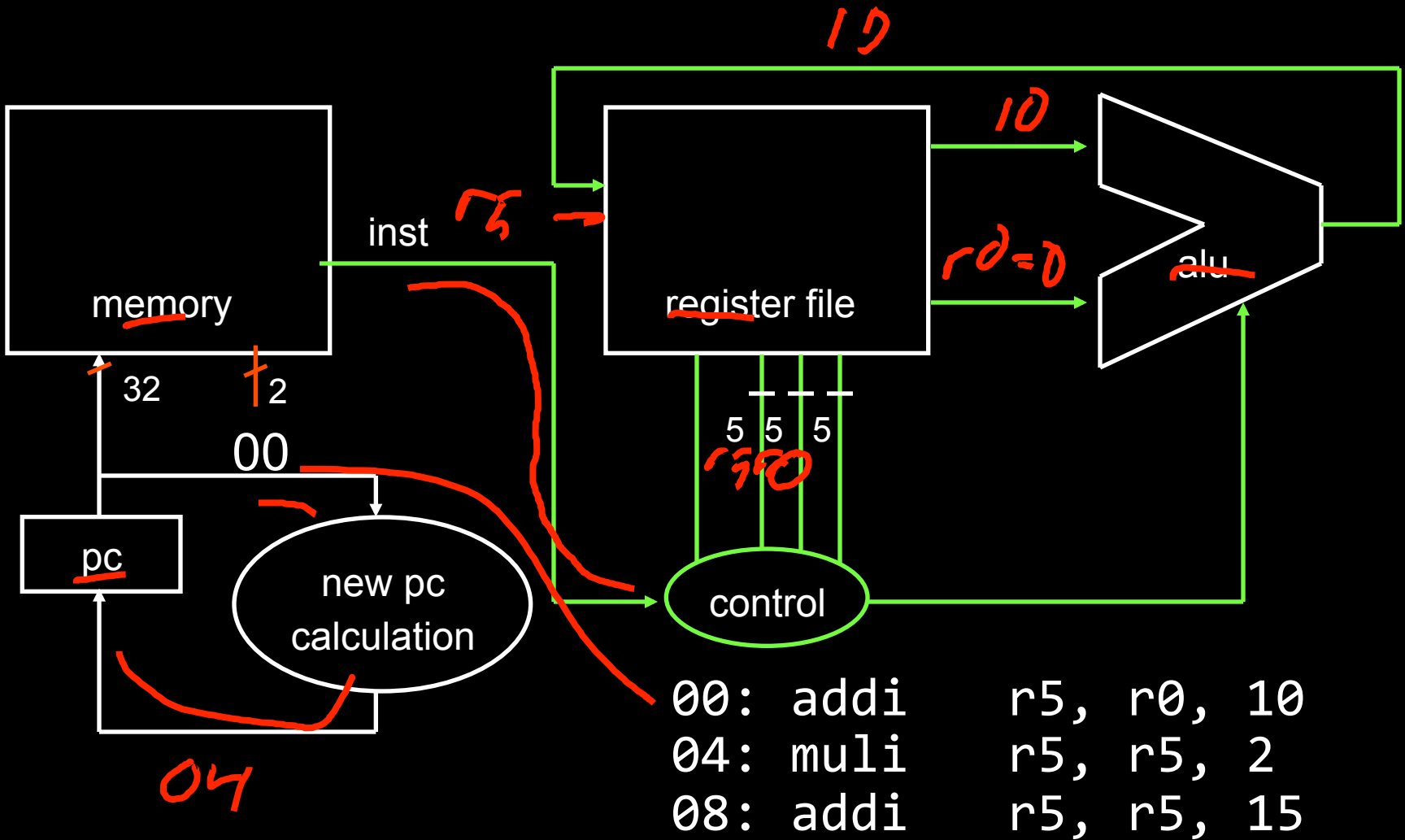


Basic Computer System

- A processor executes instructions
 - Processor has some internal state in storage elements (registers)
- A memory holds instructions and data
 - von Neumann architecture: combined inst and data
- A bus connects the two



Simple Processor



Inside the Processor

- AMD Barcelona: 4 processor cores

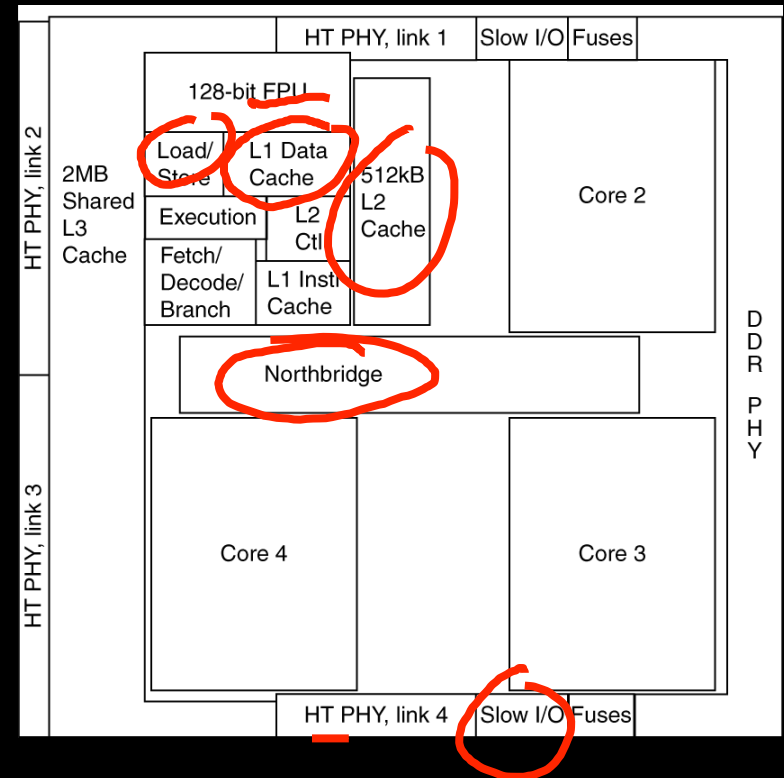
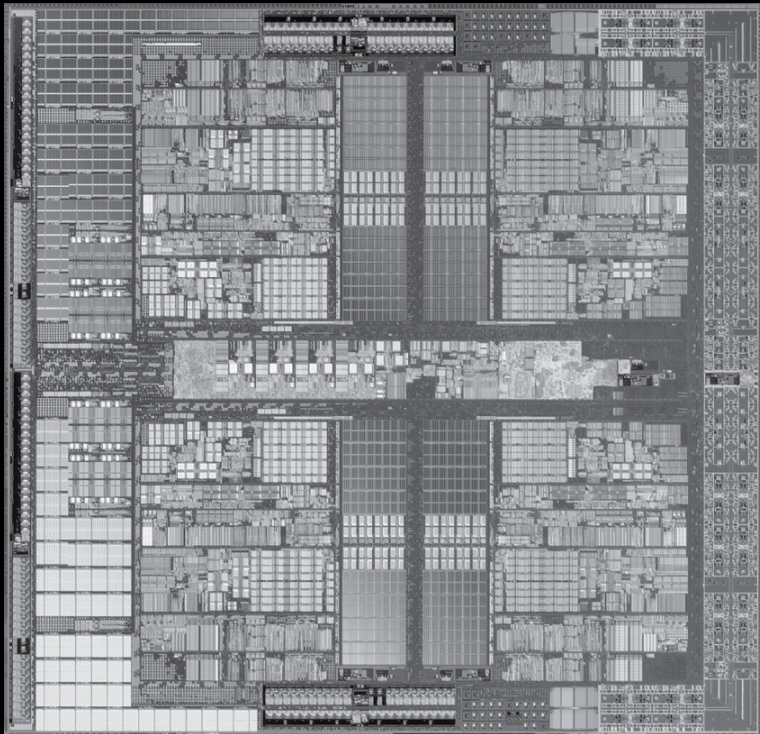
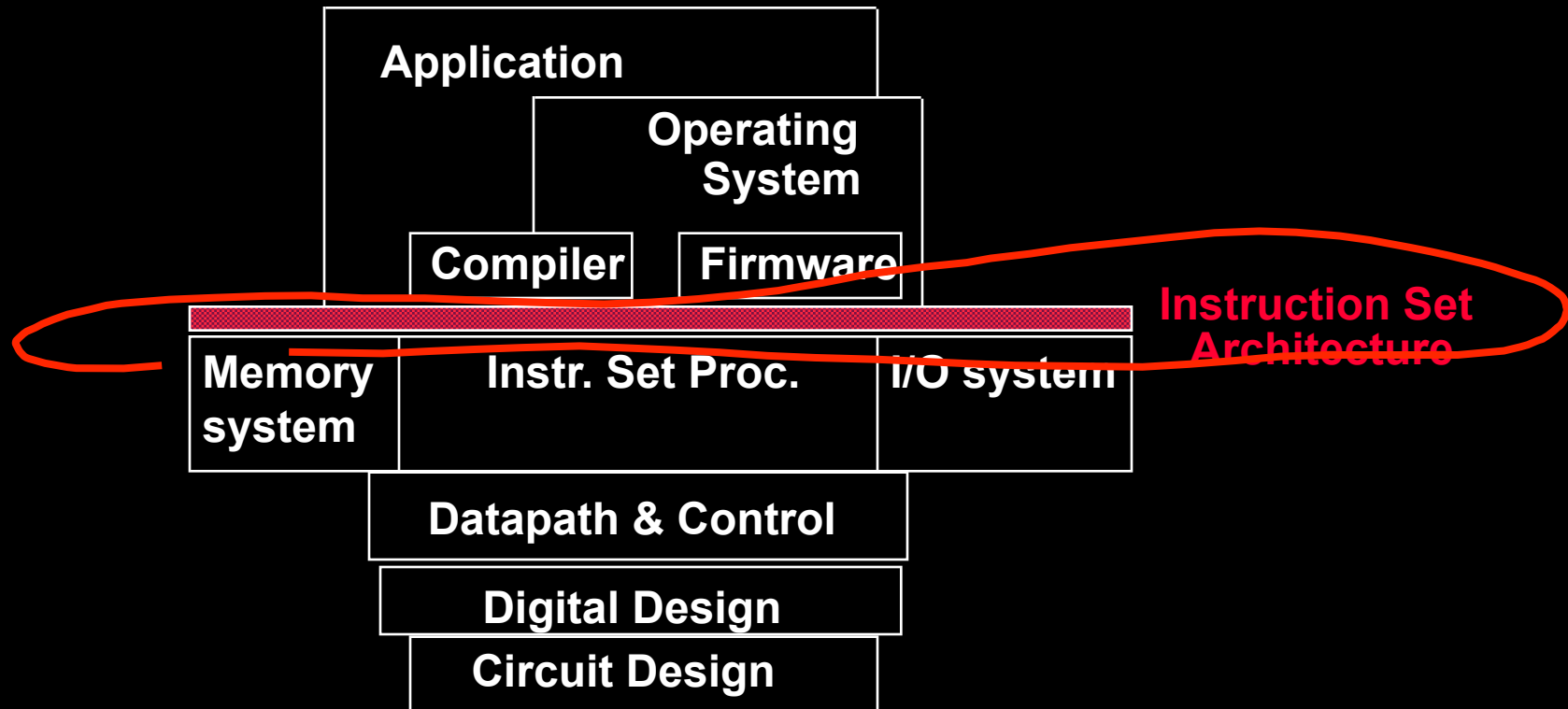


Figure from Patterson & Hennessy, Computer Organization and Design, 4th Edition

Overview

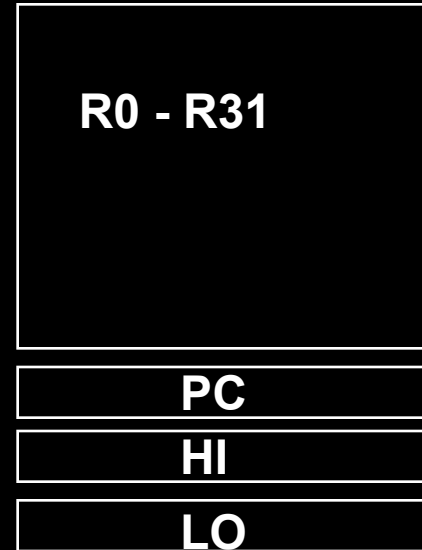


MIPS R3000 ISA

- Instruction Categories

- Load/Store
- Computational
- Jump and Branch
- Floating Point
 - coprocessor
- Memory Management

Registers



Calling Conventions

Labels

```
main:  
  jal mult  
Laftercall1:  
  add $1,$2,$3  
  
  jal mult  
Laftercall2:  
  sub $3,$4,$5
```

```
mult:  
  addiu sp,sp,-4  
  sw $31, 0(sp)  
  beq $4, $0, Lout  
  ...  
  jal mult  
Linside:  
  ...  
Lout:  
  lw $31, 0(sp)  
  addiu sp,sp,4  
  jr $31
```

Data Layout



```
blue() {  
    pink(0,1,2,3,4,5);  
}
```

```
pink() {  
    orange(10,11,12,13,14);  
}
```

Buffer Overflows



```
blue() {  
    pink(0,1,2,3,4,5);  
}  
pink() {  
    orange(10,11,12,13,14);  
}  
orange() {  
    char buf[100];  
    gets(buf); // read string, no check  
}
```

Parallel Processing

- Spin Locks
- Shared memory, multiple cores
- Etc.

Applications

- Everything these days!
 - Phones, cars, televisions, games, computers,...

Why should you care?

- Bridge the gap between hardware and software
 - How a processor works
 - How a computer is organized
- Establish a foundation for building higher-level applications
 - How to understand program performance
 - How to understand where the world is going

Example: Can answer the question...

- A: for $i = 0$ to 99
 - for $j = 0$ to 999
 - $A[i][j] = \text{complexComputation}()$
- B: for $j = 0$ to 999
 - for $i = 0$ to 99
 - $A[i][j] = \text{complexComputation}()$
- Why is B 15 times slower than A?

Example 2: Moore's Law

The number of transistors integrated on a single die will double every 24 months...

– Gordon Moore, Intel co-founder, 1965

Amazingly Visionary

1971 – 2300 transistors – 1MHz – 4004

1990 – 1M transistors – 50MHz – i486

2001 – 42M transistors – 2GHz – Xeon

2004 – 55M transistors – 3GHz – P4

2007 – 290M transistors – 3GHz – Core 2 Duo

2009 – 731M transistors – 2GHz – Nehalem

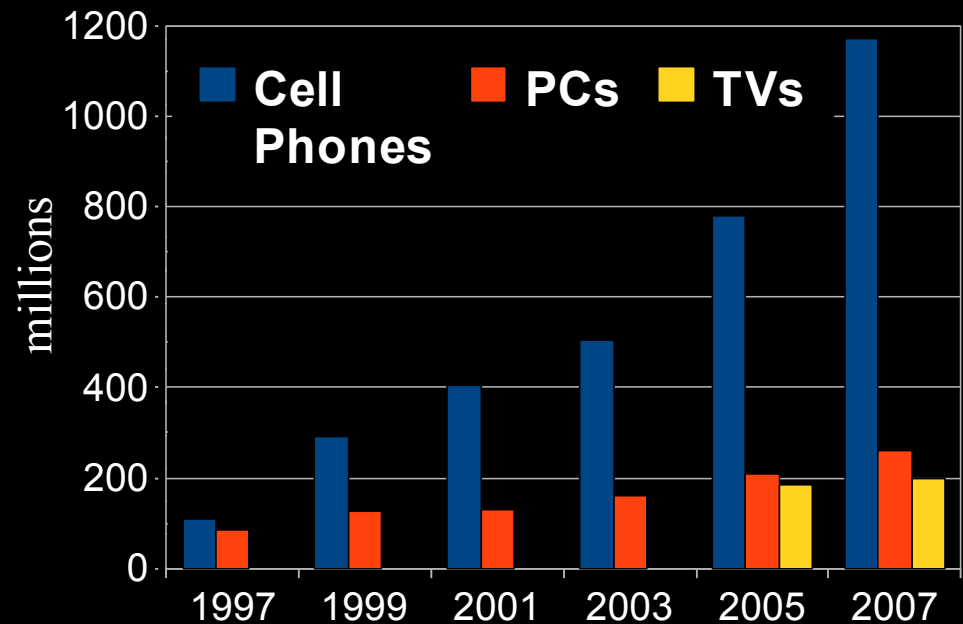
Example 3: New Devices



Xilinx FPGA

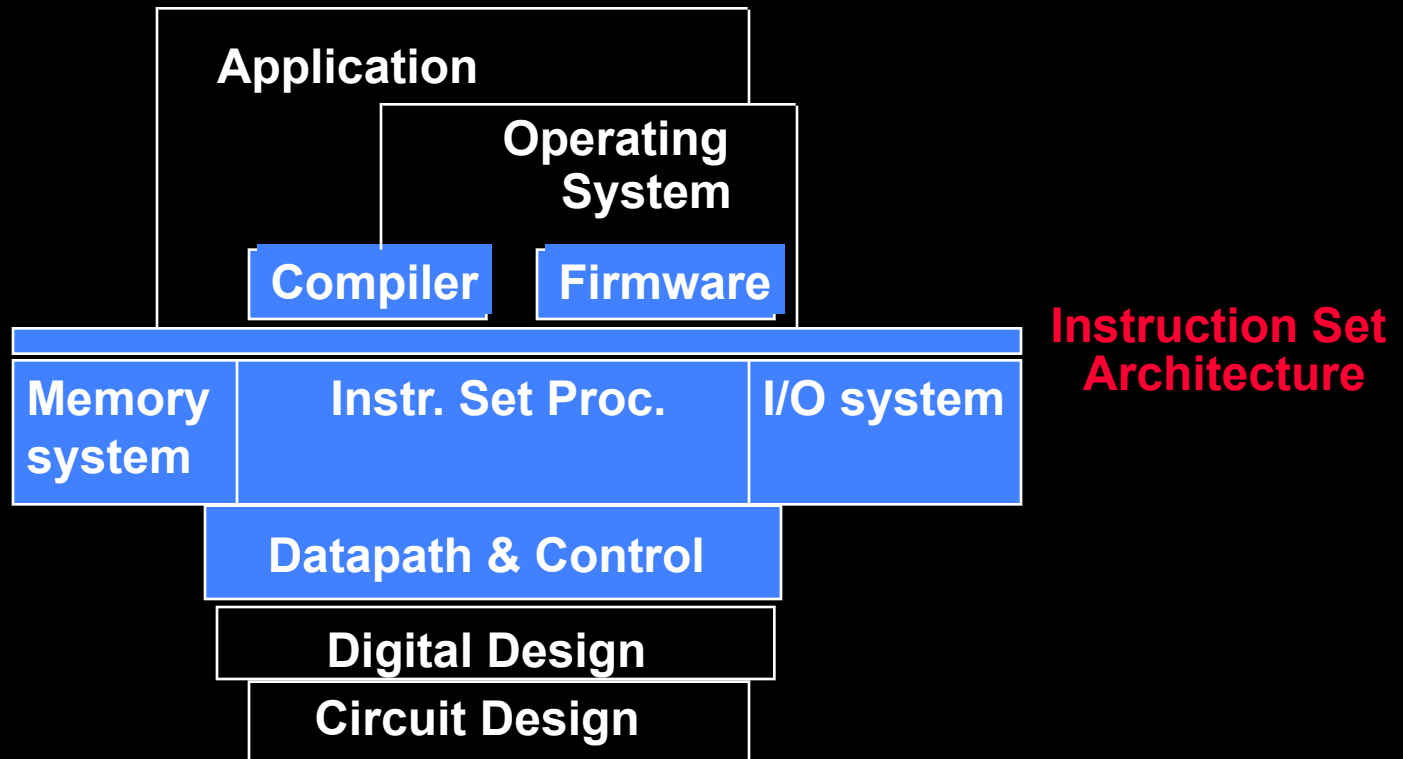


NVidia GPU



Berkeley mote

Covered in this course

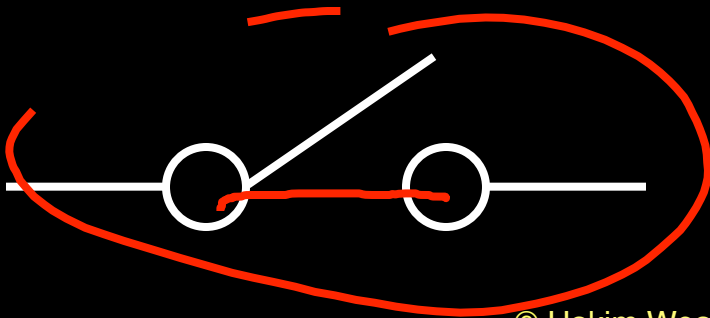


Nuts and Bolts: Switches, Transistors, Gates

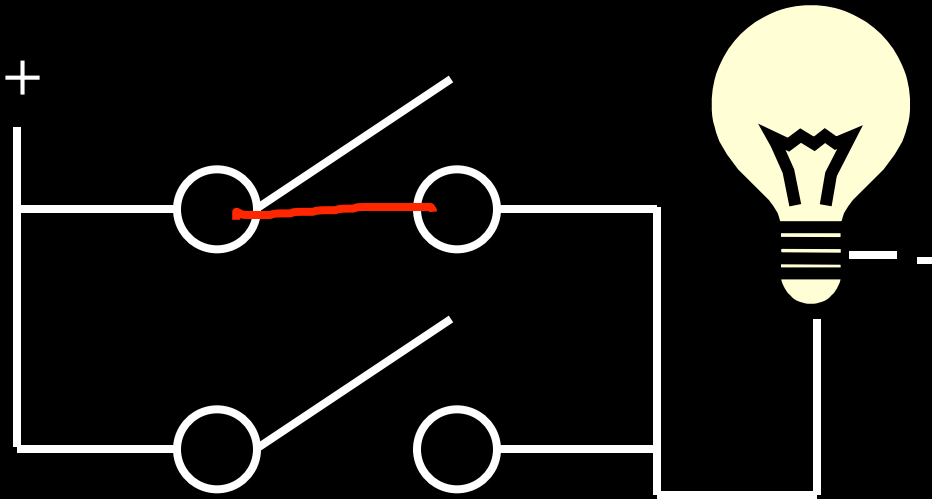
A switch



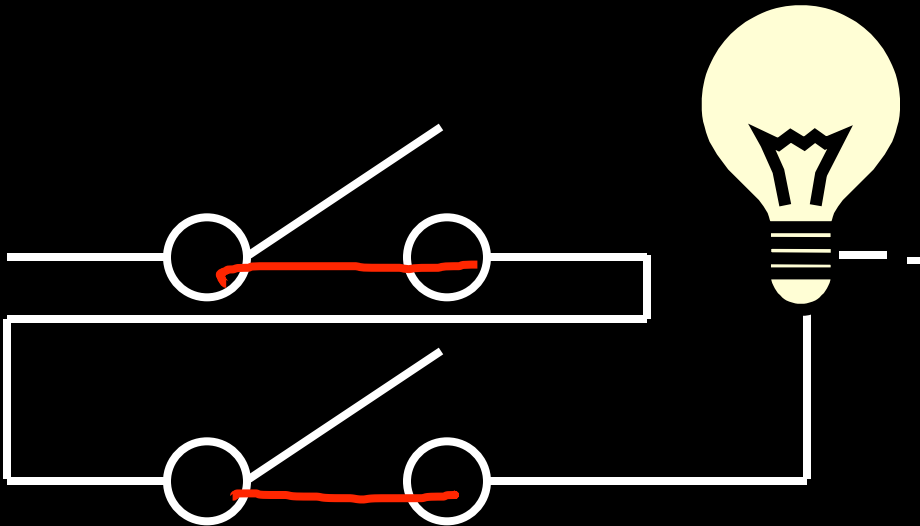
- A switch is a simple device that can act as a conductor or isolator
- Can be used for amazing things...



Switches



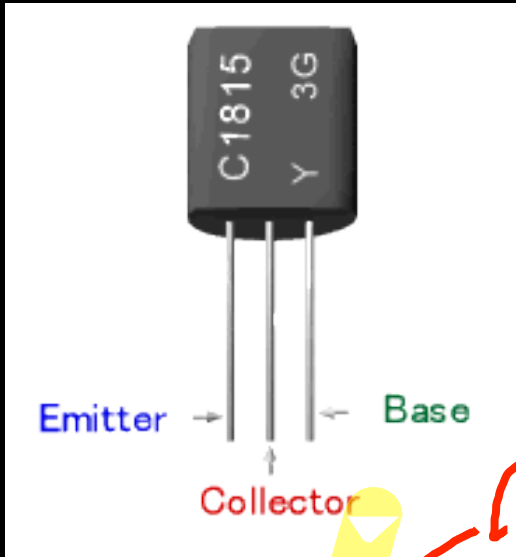
- Either (OR)



- Both (AND)

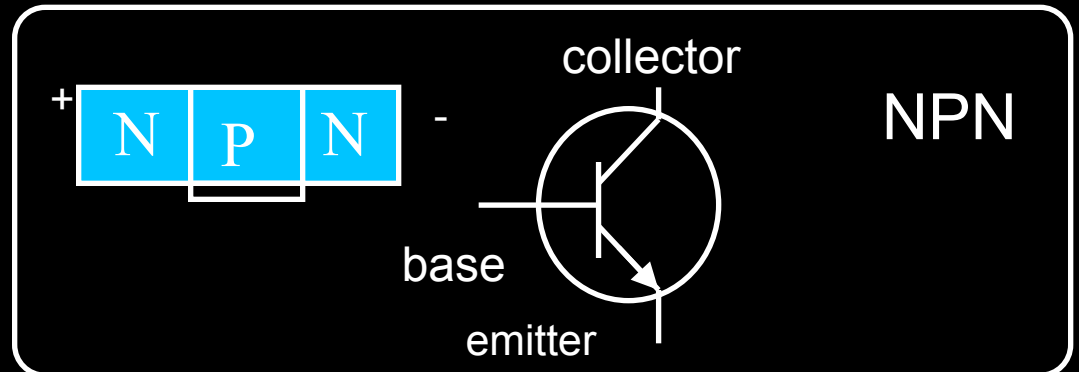
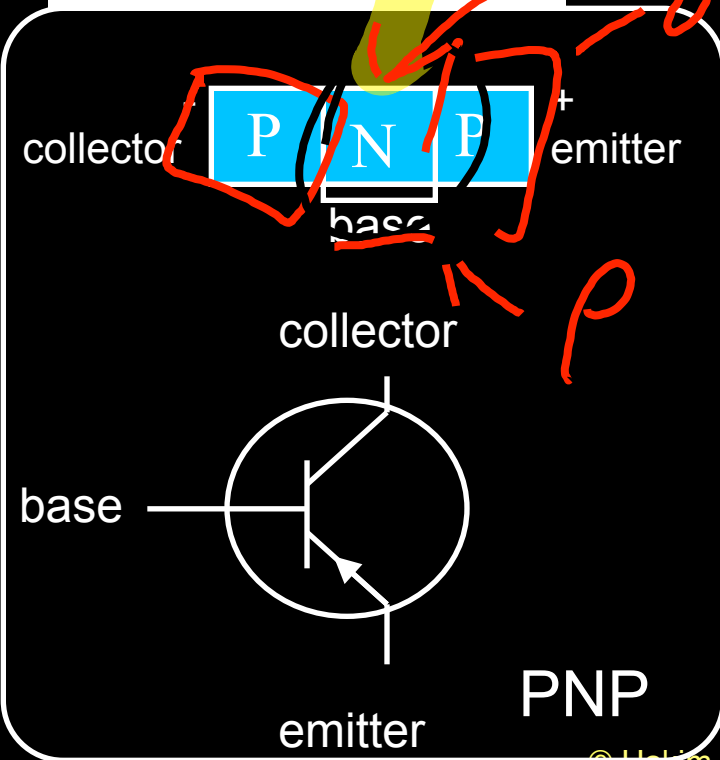
- But requires mechanical force

Transistors



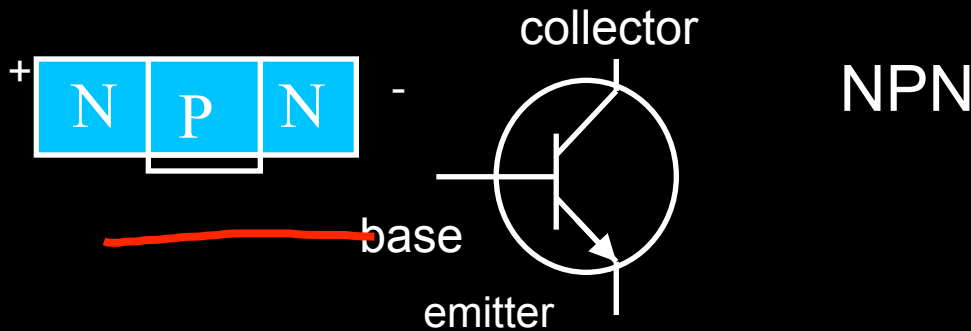
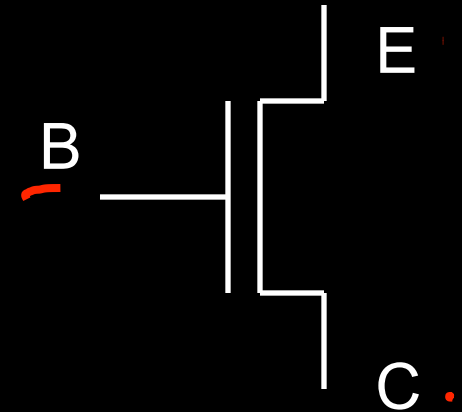
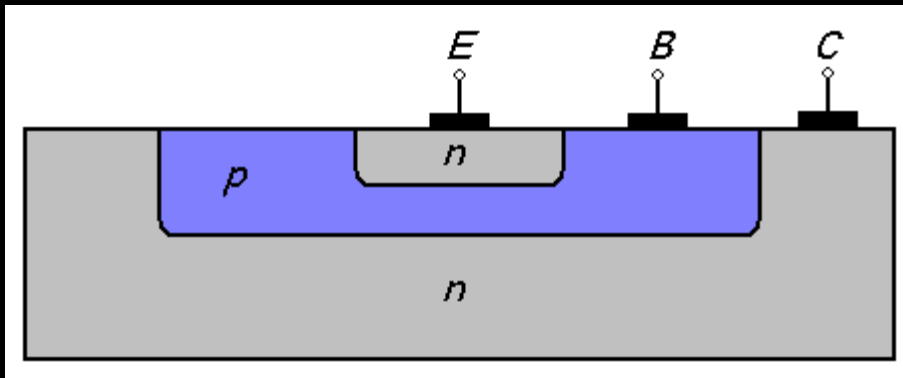
- Solid-state switch
 - The most amazing invention of the 1900s

- PNP and NPN



NPN Transistors

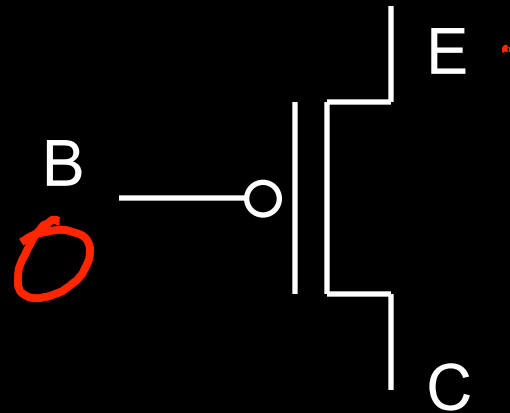
- Semi-conductor



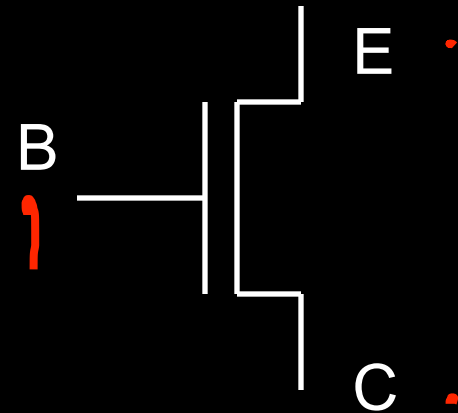
- Connect E to C when base = 1

P and N Transistors

- PNP Transistor



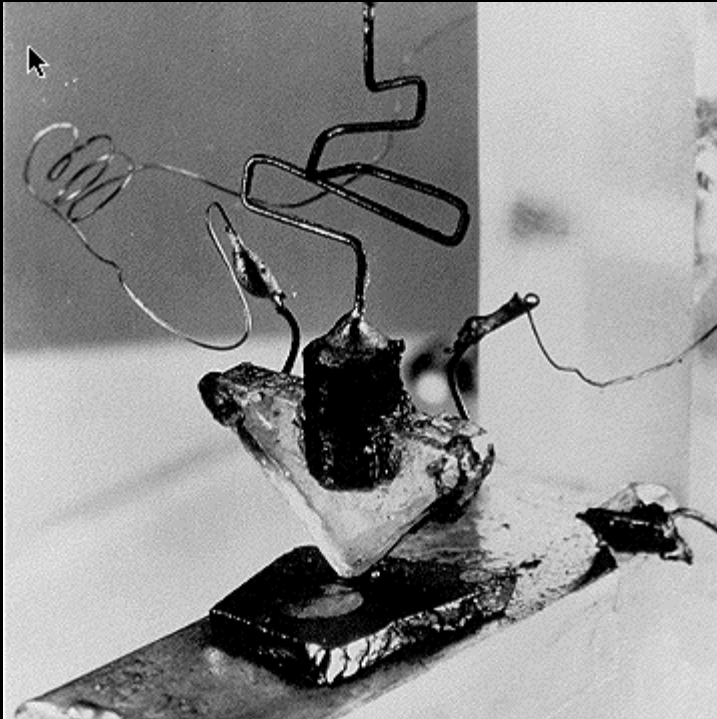
- NPN Transistor



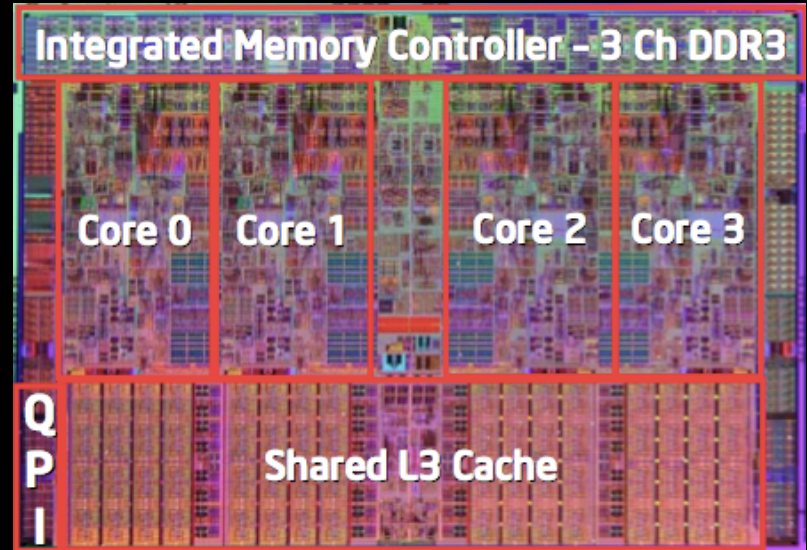
- Connect E to C when base = 0

- Connect E to C when base = 1

Then and Now

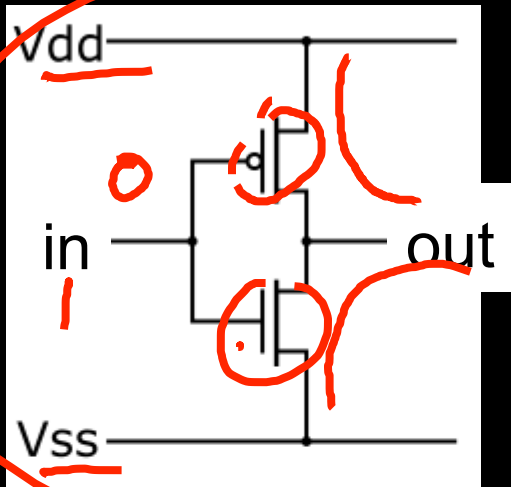


- The first transistor
 - on a workbench at AT&T Bell Labs in 1947

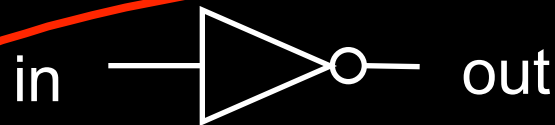


- An Intel Nehalem
 - 731 million transistors

Inverter



- Function: NOT
- Called an inverter
- Symbol:

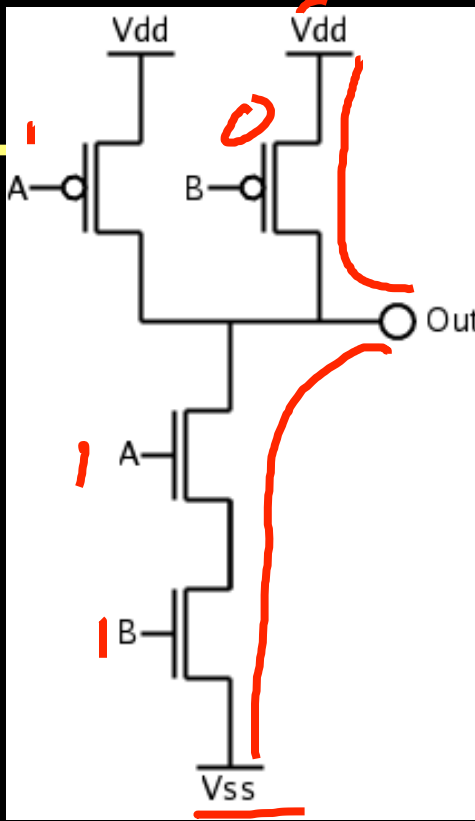


In	Out
0	1
1	0

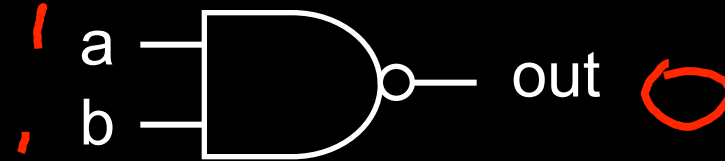
Truth table

- Useful for taking the inverse of an input
- CMOS: complementary-symmetry metal-oxide-semiconductor

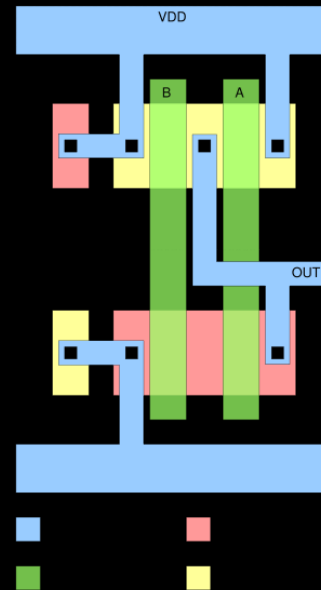
NAND Gate



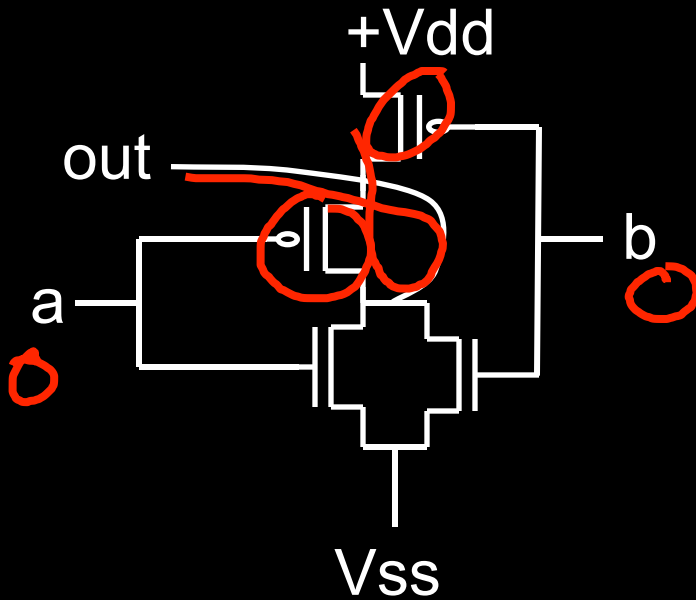
- Function: NAND
- Symbol:



A	B	out
0	0	1
1	0	1
0	1	1
1	1	0

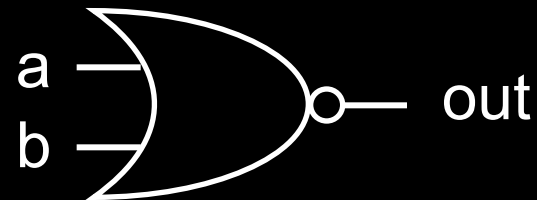


NOR Gate

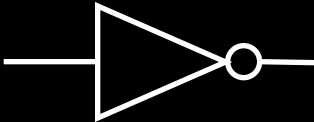
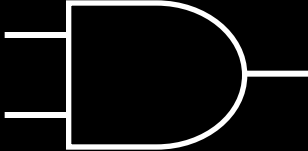
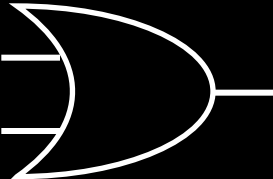


- Function: NOR
- Symbol:

A	B	out
0	0	1
1	0	0
0	1	0
1	1	0



Building Functions

- NOT: A NOT gate symbol, consisting of a triangle pointing to the right with a small circle at its tip. It has one input line on the left and one output line on the right.
- AND: An AND gate symbol, which is a D-shaped gate with two input lines on the left and one output line on the right.
- OR: An OR gate symbol, which is a D-shaped gate with a curved left side and one output line on the right. It has two input lines on the left.
- NAND and NOR are universal
 - Can implement any function with NAND or just NOR gates
 - useful for manufacturing

Reflect

Why take this course?

- **Basic knowledge needed for *all* other areas of CS:**
operating systems, compilers, ...
- **Levels are not independent**
hardware design ↔ software design ↔ performance
- **Crossing boundaries is hard but important**
device drivers
- **Good design techniques**
abstraction, layering, pipelining, parallel vs. serial, ...
- **Understand where the world is going**