CMP 334 (1/28/19)

Course syllabus (part 1)

- Course prerequisites

Course overview

Abstraction: models and interfaces

Basic computer model: the **TOY** computer

Computer design (the big picture)

Digital logic circuits

- Logisim (http://www.cburch.com/logisim/)

Numbers

Course Information

Instructor:	Bowen Alpern	
Office:	GI 137-A	
Hours:	M W 5:00 – 6:00 pm	
Sections:	afternoon	evening
Times:	M W 3:00 – 4:40 pm	7:50 – 9:30 pm
Room:	GI 231	GI 231

Text (optional)

Computer Organization and Design – Patterson & Hennessy

Computer Organization and Architecture – Null & Lobur

Computer Organization and Architecture – Stalling

CMP 334 Computer Organization

Introduction to digital logic-expressions, gates, flip-flops, adders. busses, multiplexers Introduction to assembly language and assembly level organization - data representation, instruction formats, addressing modes, interrupts. Memory systems - caches (mapping and management policies) and memory hierarchies, latency and bandwidth, virtual memory (page tables, TLB). Input/Output- busses, channels and DMA. Performance considerations- pipelining, RISC architecture, branch prediction, introduction to instruction level parallelism.

Credits: 4

PREREQS: CMP167 CMP 232 or departmental permission.

CMP 334 Computer Organization

Introduction to digital logic-expressions, gates, flip-flops, adders. busses, multiplexers Introduction to assembly language and assembly level organization - data representation, instruction formats, addressing modes, interrupts. Memory systems - caches (mapping and management policies) and memory hierarchies, latency and bandwidth, virtual memory (page tables, TLB). Input/Output- busses, channels and DMA. Performance considerations- pipelining, RISC architecture, branch prediction, introduction to instruction level parallelism.

Credits: 4 **Expect to work 12 hours per week outside class**

PREREQS: CMP167 CMP 232 or departmental permission.

CMP 167 Programming Methods I

(AKA: Introduction to Computer Programming)

Structured **computer programming** using a modern high-level programming language. Includes console I/O, data types, variables, control structures, including iteration, arrays, function definitions and calls, parameter passing, functional decomposition, and an introduction to objects. Debugging techniques.

CMP 232 Elementary Discrete Structures & Applications to Computer Science

(AKA: Finite Mathematics)

Sets, relations, and functions; **propositional calculus, Boolean algebras, and combinatorial circuits**, counting methods; proof techniques; analysis of algorithms; graphs and trees, puzzles; finite machines, **sequential circuits**, and recognizers.

Truth table ≡ Boolean function ≡ combinational circuit

Binary numbers, change of basis

Basic High-School Algebra

Equation solving

$$\frac{6}{11} = \frac{9}{4 + 5 \cdot x} \qquad \text{what is } x\text{?}$$

Word problems

Initially Alice has twice as many apples as Bob. After she eats two apples and gives three to Carol, she still has 3 more apples than Bob. How many apples does Bob have?

Advanced Middle-School Arithmetic

Fractions

$$\frac{5}{7} + \frac{2}{3} \qquad \frac{3}{4} - \frac{6}{11} \qquad \frac{5}{8} \cdot \frac{4}{5} \qquad \frac{12}{5} \div \frac{3}{8}$$

Long division

$$56298 \,/\, 24 \;=\; 2345.75$$

Binary numbers

$$110101011001_2 \;=\; D59_{16} \;=\; 1407_{10}$$

Long Division

56298 / 24 =

$$
\begin{array}{r}
2345.75 \\
24 \,\overline{)\,56298.00} \\
\underline{-48} \\
82 \\
\underline{-72} \\
109 \\
\underline{-96} \\
138 \\
\underline{-120} \\
18.0 \\
\underline{-16.8} \\
1.20 \\
\underline{-1.20} \\
0
\end{array}
$$

Amdahl's Law Example

$$\frac{1}{0.5 \cdot \frac{2}{5} + \left(1 - \frac{2}{5}\right)}$$

Amdahl's Law Example

$$\frac{1}{0.5 \cdot \frac{2}{5} + \left(1 - \frac{2}{5}\right)} = \frac{1}{\frac{1}{5} + \frac{3}{5}}$$

Amdahl's Law Example

$$\frac{1}{0.5 \cdot \frac{2}{5} + \left(1 - \frac{2}{5}\right)} = \frac{1}{\frac{1}{5} + \frac{3}{5}}$$

$$= \frac{1}{\frac{4}{5}}$$

Amdahl's Law Example

$$\frac{1}{0.5 \cdot \frac{2}{5} + \left(1 - \frac{2}{5}\right)} = \frac{1}{\frac{1}{5} + \frac{3}{5}}$$

$$= \frac{1}{\frac{4}{5}}$$

$$= \frac{5}{4}$$

Amdahl's Law Example

$$\frac{1}{0.5 \cdot \frac{2}{5} + \left(1 - \frac{2}{5}\right)} = \frac{1}{\frac{1}{5} + \frac{3}{5}}$$

$$= \frac{1}{\frac{4}{5}}$$

$$= \frac{5}{4}$$

$$= 1.2$$

Course Overview

ALU (Arithmetic / Logical Unit) → *exam 1* (3/13/19)

- Binary arithmetic (unsigned and signed numbers)
- Combinational circuit ≡ Boolean function ≡ truth table
- Assembly language programming (part 1)

CPU (Central Processing Unit) → *exam 2* (4/17/19)

- Sequential circuits: latches, flip-flops, processors
- Implicit parallelism: pipeline processors and ILP
- Assembly language programming (part 1)

Computer → *final* ????

- The memory hierarchy
- Explicit parallelism

Course Overview (part 1: ALU)

Binary arithmetic (unsigned and signed numbers)

- Binary, decimal, and hexadecimal numbers
- Addition, subtraction, comparisons, ...

Combinational circuits (truth tables, Boolean functions)

- Combinational circuit design process
- Inverters, decoders, multiplexors
- Adders, 1-bit (half and full), n-bit (ripple-carry); ALU

Assembly language programming, part 1

- ALU instructions: addition, subtraction, …
- Data transfer instructions: load and store
- Subprograms: function call and return

Recurring Themes

Moore's law

Abstraction to manage complexity

Models — simplification of design

Interfaces — separation of concerns

Focus on ***common case*** performance

Parallelism

Implicit: pipelined processor

Explicit: multiprocessors

The ***memory hierarchy***

Abstraction 1: Models

Model – simplified understanding of a mechanism

- Presents key features
- Ignores less relevant details

Basic Computer Model (the TOY computer)

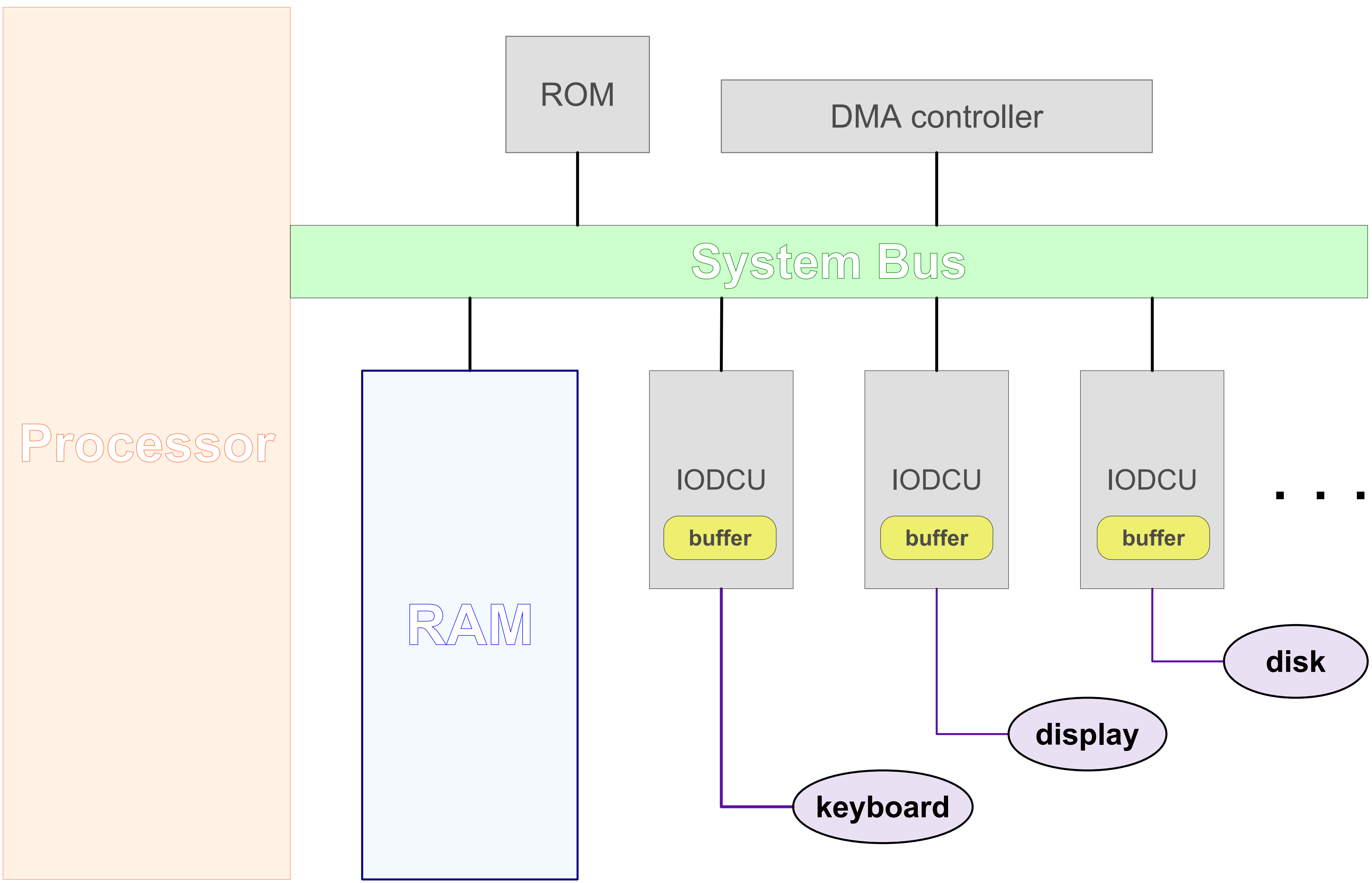
- Main computer components and their relationships
  - Processor, memory, system bus, I/O devices

Basic Processor Model (the TOY processor)

- Main processor components and their relationships
  - ALU, registers (GP, PC, IR), buses (A, B, C)

We will refine these models over the semester

The **TOY** Computer

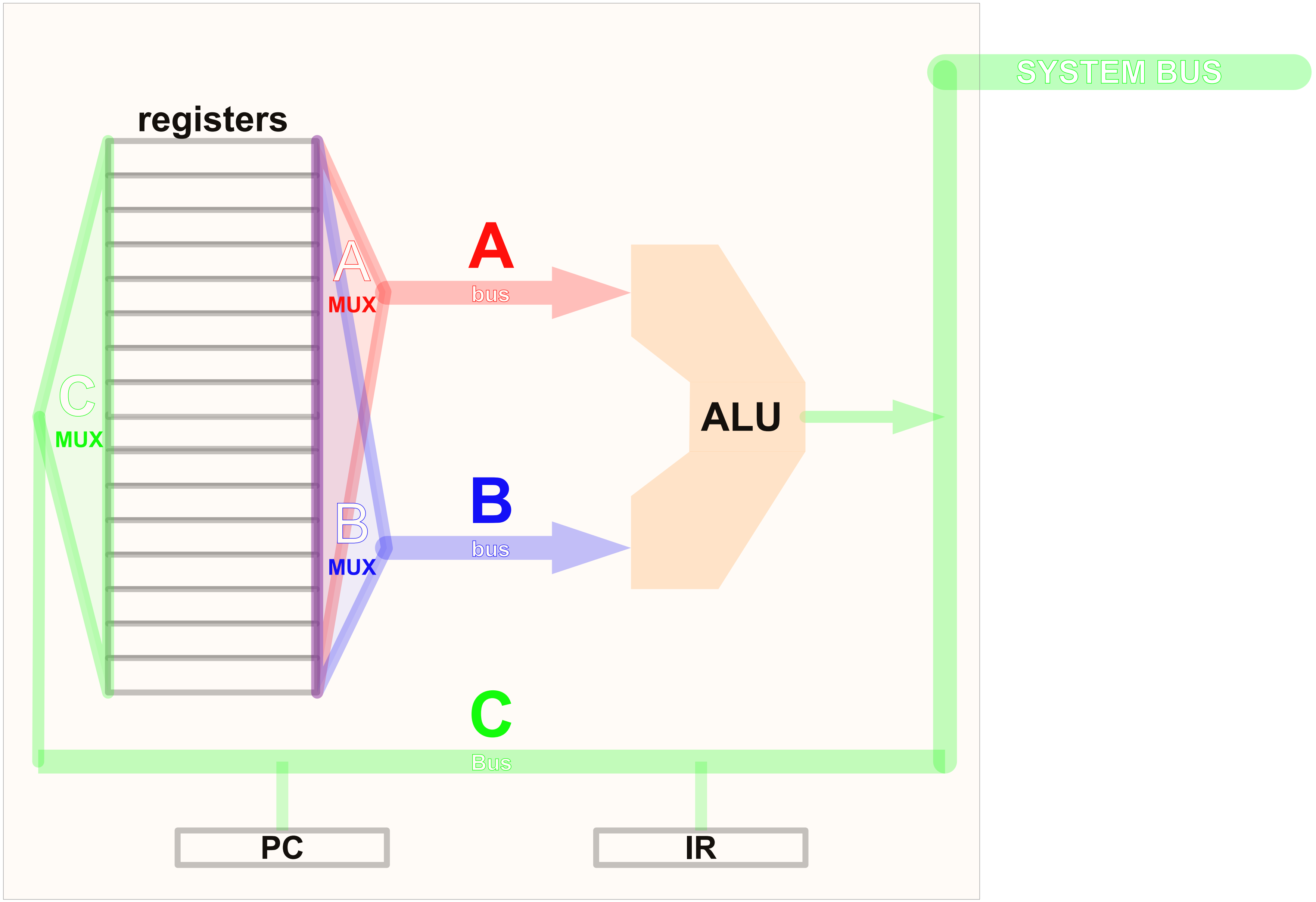


HW 1: Computer Components

In two or three sentences of your own words define, describe, or discuss the following components of the TOY computer:

1. The System Bus
2. ROM Memory
3. RAM Memory
4. IODCU
5. IODCU buffer
6. The Processor
7. The DMA Controller

The **TOY** Processor



Abstraction 2: Interfaces

Interface – <u>separation of concerns</u>

- Boundary – between objects or systems
- Protocol – rules for interaction between parties
- Contract – formalized expectations

Distribution of Labor

- User (consumer) ignores *implementation*
- Provider (producer) ignores *application*

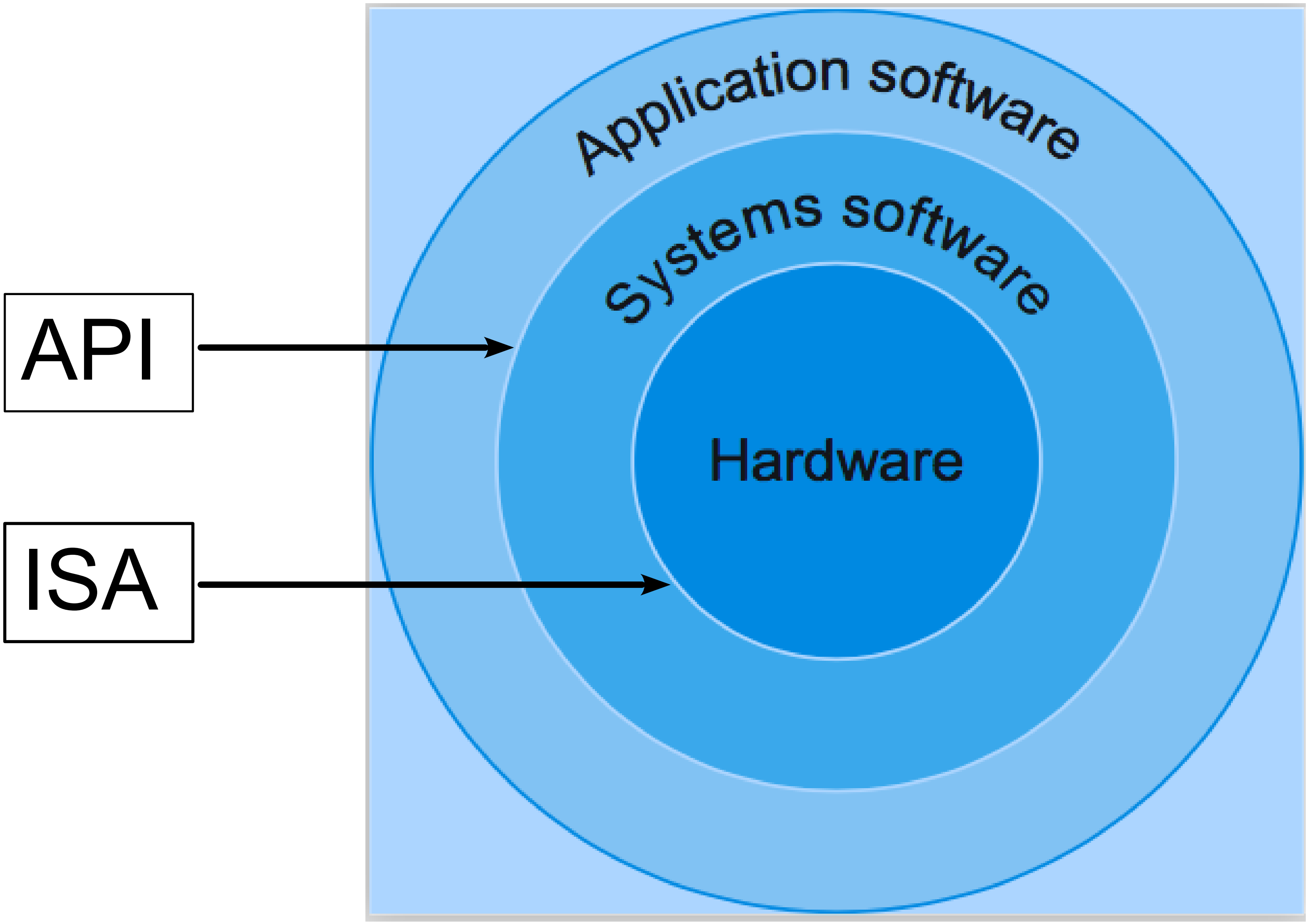
Instruction **S**et **A**rchitecture **(ISA)**

- Between hardware & software

Application **P**rogram **I**nterface **(API)**

- Between application program & operating system

Computer Interface Diagram



What Happens to Your Program

High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

Compiler

Assembly language program (for ARMv8)

```
swap:
      LSL  X10, X1,3
      ADD  X10, X0,X10
      LDUR X9, [X10,0]
      LDUR X11,[X10,8]
      STUR X11,[X10,0]
      STUR X9, [X10,8]
      BR   X10
```

Assembler

Binary machine language program (for ARMv8)

```
000000001010001000000000100011000
00000000100000100001000000100001
10001101111000100000000000000000
10001110000100100000000000000100
10101110000100100000000000000000
10101101111000100000000000000100
00000011111000000000000000001000
```

Computer Design – The Big Picture

A computer is one big ***sequential*** circuit

Abstract into discrete *sequential* components:

Combinational circuits + memory + clock

Combinational circuit design process:

1. Specify semantics

   Black box means that it cannot see what it is inside

   a. Black Box diagram (identifies input and output)

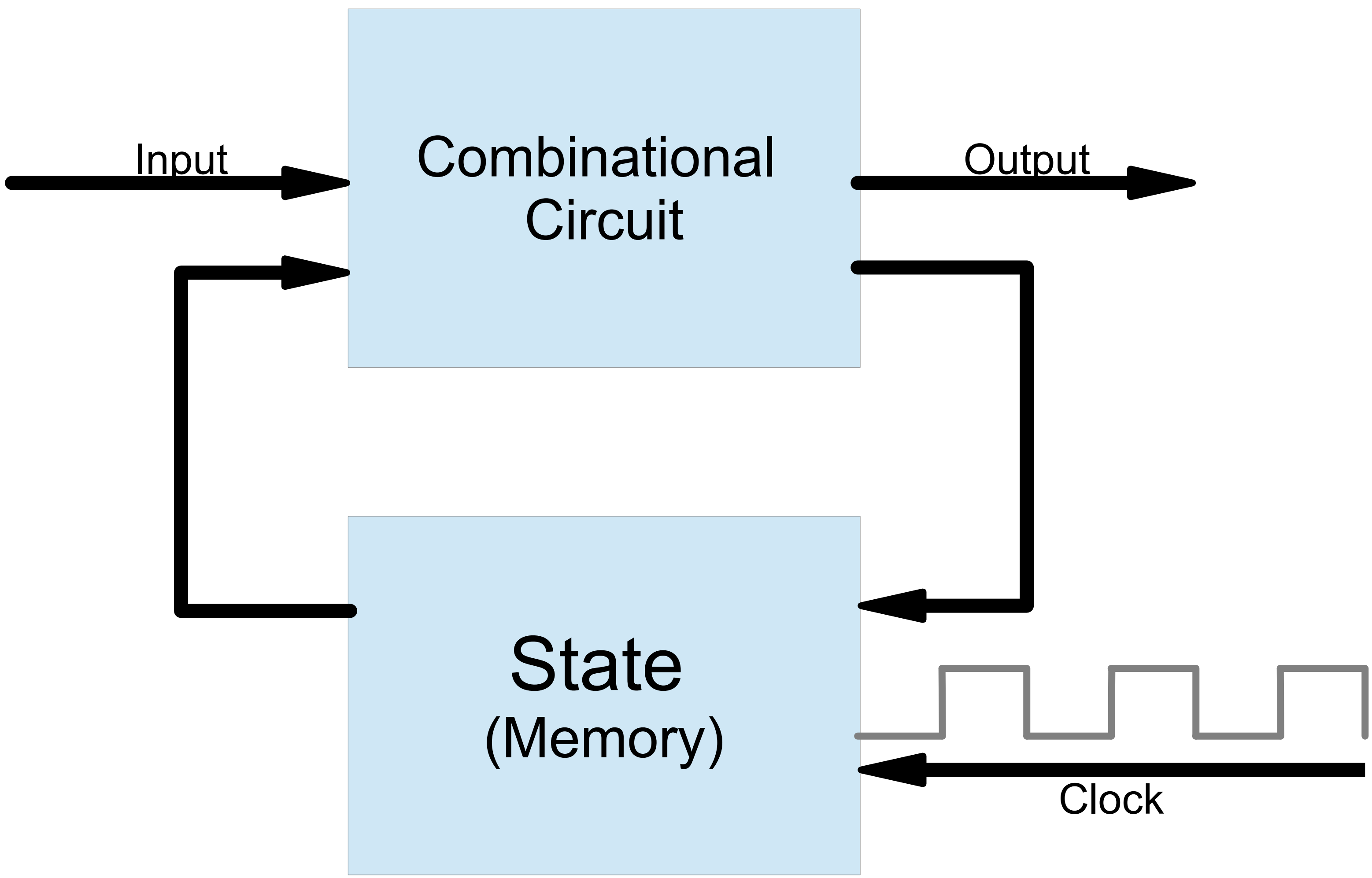
   b. *Truth Table* (input determines output)

2. Truth table → *Boolean formula*

3. Minimize boolean formula (Karnaugh Maps)

4. Boolean formula → combinational circuit

Synchronous Sequential Circuit

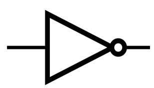
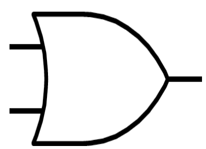
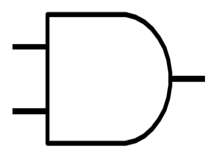


Digital Logic Circuit

Directed graph with labeled nodes

- Edges are called ***wires***
- Nodes are called ***gates***

Some gate labels *restrictions*

not 		in–degree = 1
or 		in–degree > 1
and 		in–degree > 1
...		
input	[constant (**0** or **1**) or identifier]	in–degree = 0
output	[identifier]	out–degree = 0

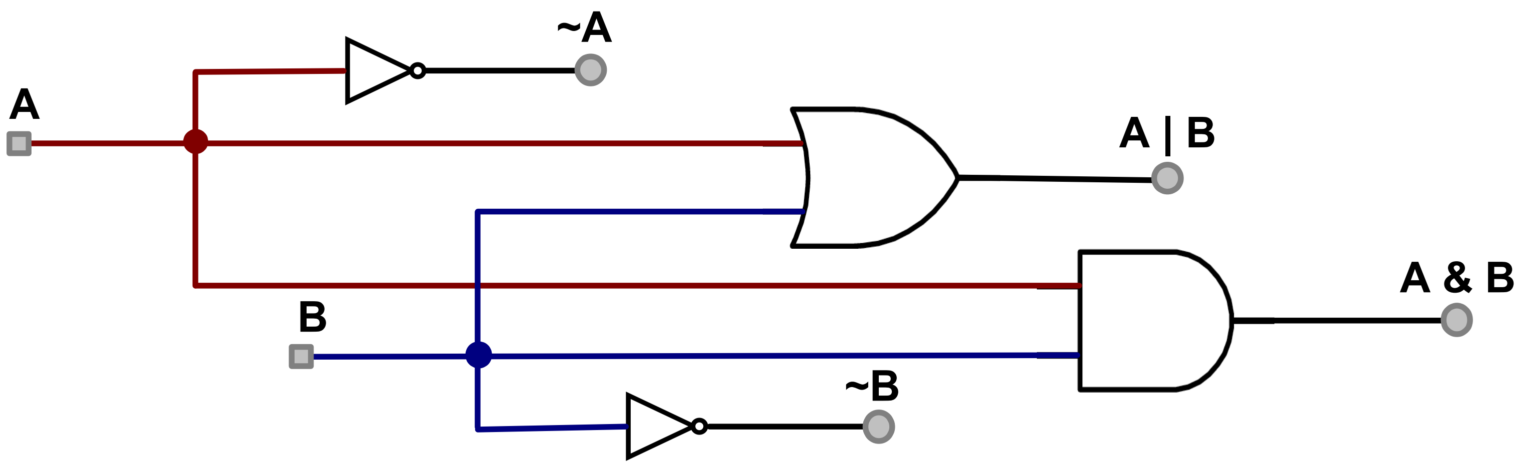
Combinational Circuit

- Digital logic circuit that is ***acyclic***
  - **No feedback loops** in the circuit
  - <u>Input determines output</u>
- Representing numeric values
  - Binary values (**0** or **1**) associated with gates & wires
    - Input (and output) variables can only hold 1 bit
  - $\mathbf{n}$ (a k-bit input) is represented by k input variables:
    - n_{k-1}, n_{k-2} … n_1, and n_0 where

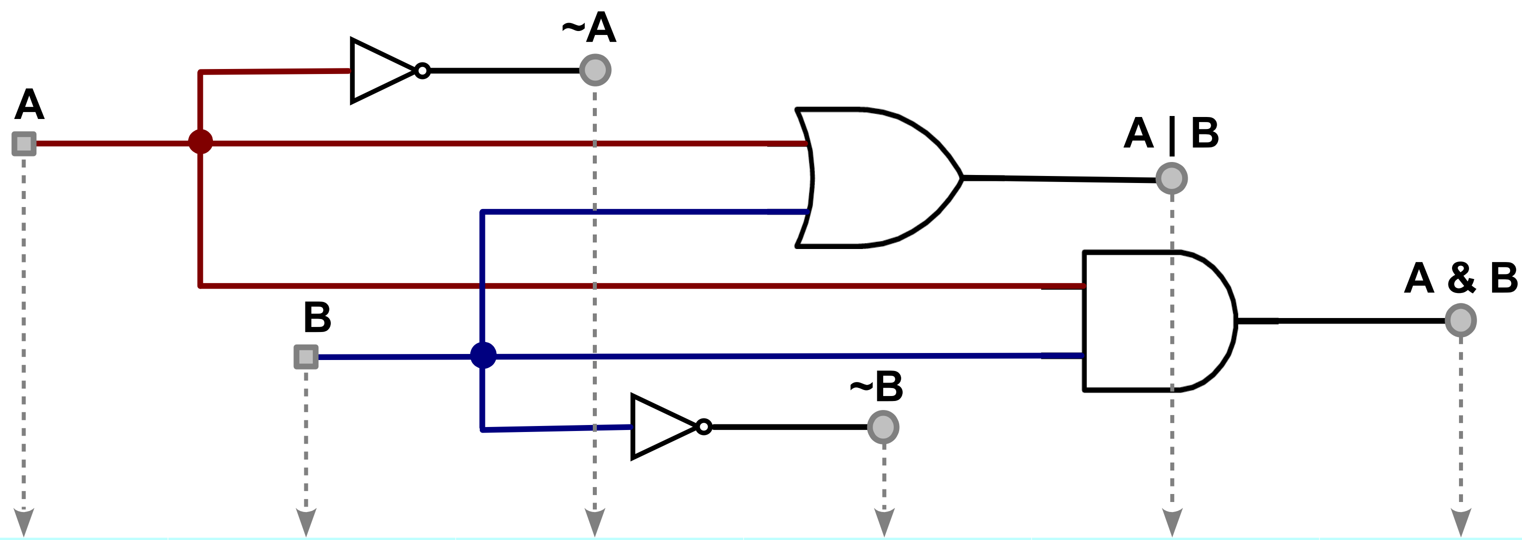
$$\mathbf{n} = \sum_{i=0}^{k-1} n_i 2^i$$

  - A k-bit ***bus*** is a bundle of k wires
    - Can be used to transmit k-bit numbers

Logic Gates

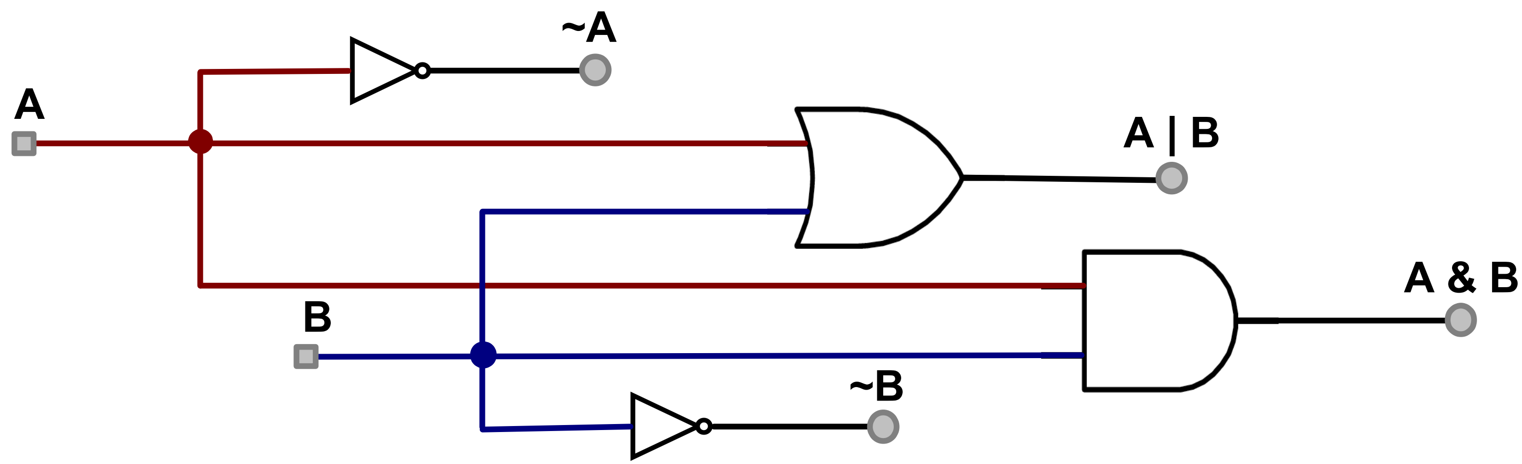


Logic Gate Truth Table



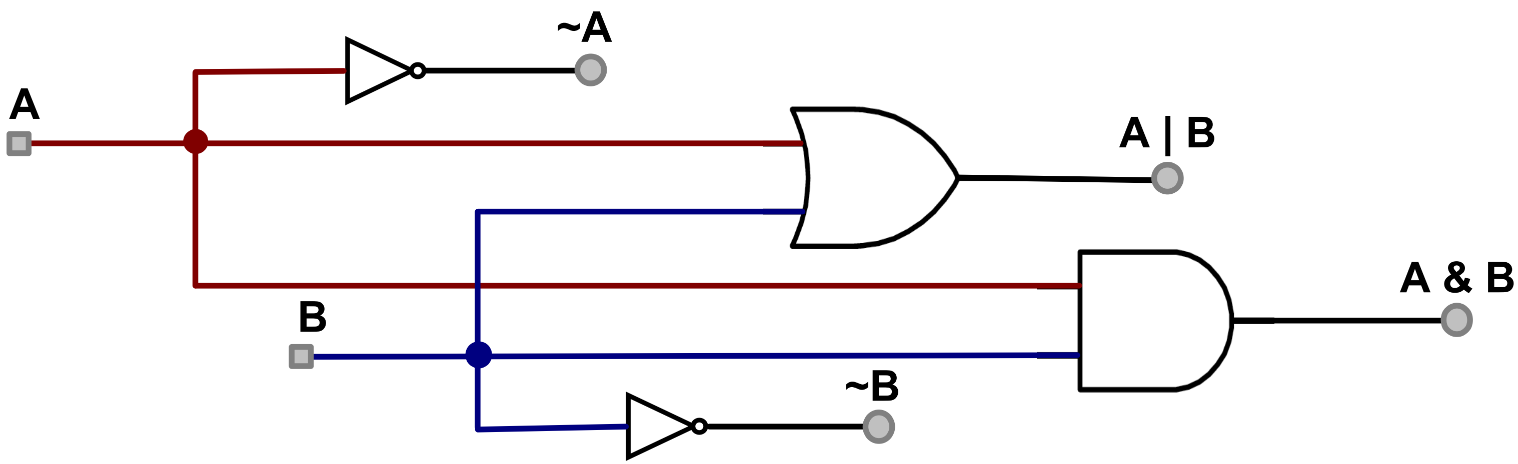
A	B	¬A	¬B	A∨B	A∧B
0	0				
0	1				
1	0				
1	1				

Logic Gate Truth Table



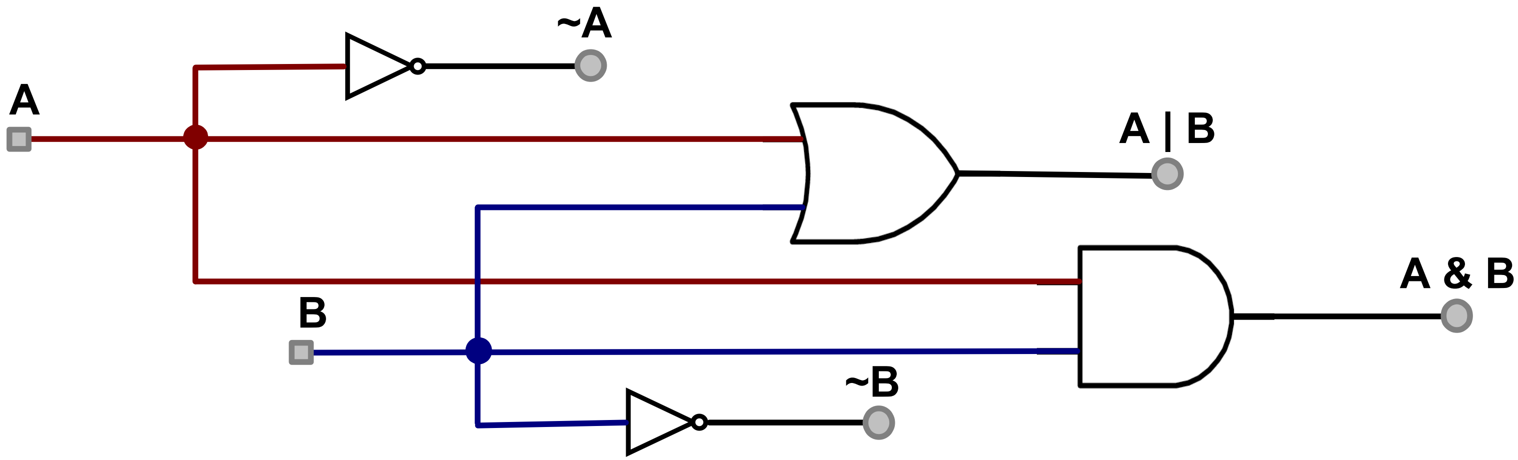
A	B	¬A	¬B	A∨B	A∧B
0	0	1			
0	1	1			
1	0	0			
1	1	0			

Logic Gate Truth Table



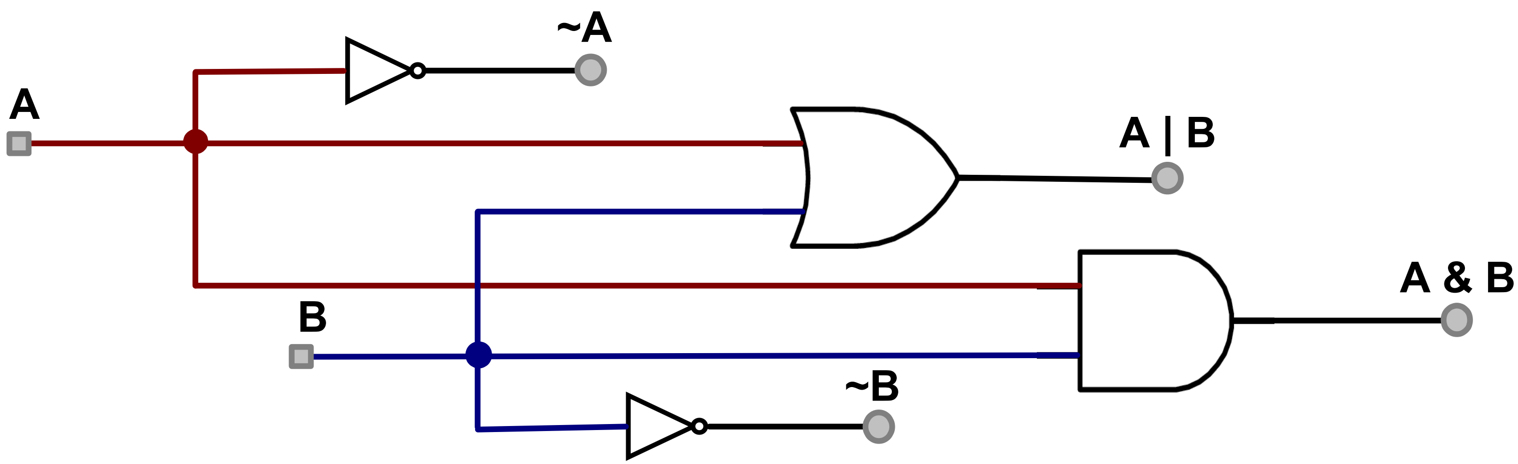
A	B	¬A	¬B	A∨B	A∧B
0	0	1	1		
0	1	1	0		
1	0	0	1		
1	1	0	0		

Logic Gate Truth Table



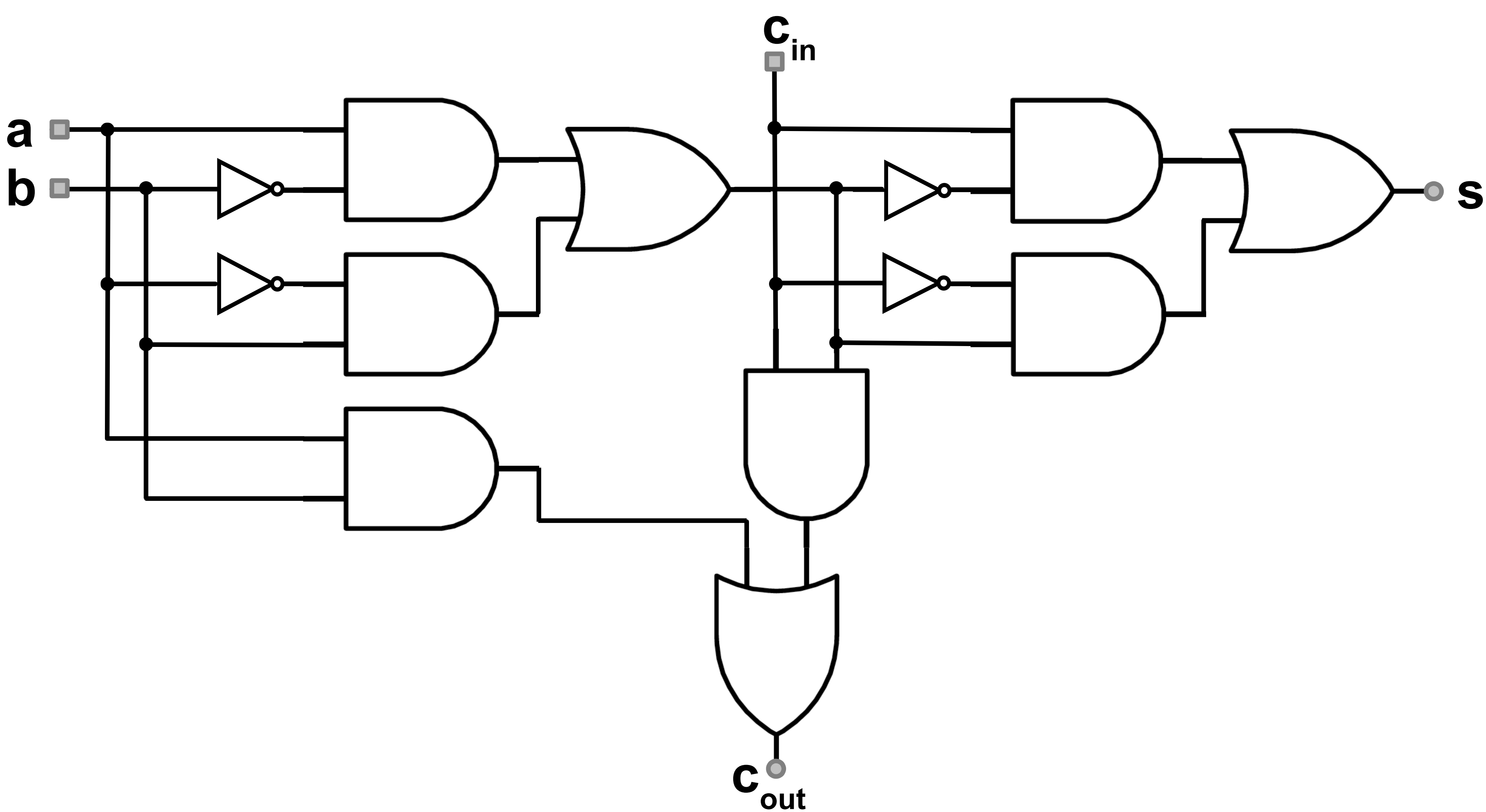
A	B	¬A	¬B	A∨B	A∧B
0	0	1	1	0	
0	1	1	0	1	
1	0	0	1	1	
1	1	0	0	1	

Logic Gate Truth Table



A	B	¬A	¬B	A∨B	A∧B
0	0	1	1	0	0
0	1	1	0	1	0
1	0	0	1	1	0
1	1	0	0	1	1

An Example Circuit



Example Circuit Truth Table

a	b	c_{in}	s	c_{out}
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Example Circuit Truth Table

a	**b**	**c_{in}**	**s**	**c_{out}**
0	0	0	**0**	**0**
0	0	1	**1**	**0**
0	1	0	**1**	**0**
0	1	1	**0**	**1**
1	0	0	**1**	**0**
1	0	1	**0**	**1**
1	1	0	**0**	**1**
1	1	1	**1**	**1**

Numbers

Natural	**0, 1, 2, 3, 4 …**
Integer	**… −4, −3, −2, −1, 0, 1, 2, 3, 4 …**
Rational (fraction)	$-\frac{1}{8}, \quad \frac{1}{3}, \quad \frac{3}{2}$
Rational (decimal)	$-0.25,\ 0.3\overline{3},\ 1.5,\ 2.1 \cdot 10^7$
Irrational	$\pi, \quad \sqrt{2}, \quad -\sqrt[3]{17}, \quad e^{3^{\frac{1}{2}}}$
Complex	$i, \quad -3i, \quad 2+7i, \quad 2-7i$
Quaternion	$1+i-j+k, \quad -2+4i+3j-k$

Numbers for Computers

Integers (fixed size, binary)

Unsigned (non-negative)

11001001_2 (201)

Signed (two's complement)

11001001_2 (–73)

00110111_2 (+73)

Floating-Point (fixed size, binary)

$+1.11001001 \cdot 2^{+0111}$

$+1.00011111 \cdot 2^{-0101} + 1.10101010 \cdot 2^{+0111}$ ¿

Natural Numbers **N**

N – a set containing 0, and s : **N** → **N** such that

1) $s(x) \neq 0$ for all x in **N**
2) $s(x) = s(y) \Rightarrow x = y$ for all x and y in **N**
3) If **A** subset of **N** such that
   0 is in **A**, and
   x in **A** => s(x) in **A**

   Then **A** = **N**

N *closed* under addition and multiplication

N *not closed* under subtraction and division

N = {0, 1, 2, 3, 4, 5, 6, 7, . . . }

The First 16 Natural Numbers

		0	0b**0000**	0x**0**	*zero*
|	i	**1**	0b**0001**	0x**1**	*one*
||	ii	**2**	0b**0010**	0x**2**	*two*
|||	iii	**3**	0b**0011**	0x**3**	*three*
||||	iv	**4**	0b**0100**	0x**4**	*four*
𝍸	v	**5**	0b**0101**	0x**5**	*five*
𝍸 |	vi	**6**	0b**0110**	0x**6**	*six*
𝍸 ||	vii	**7**	0b**0111**	0x**7**	*seven*
𝍸 |||	viii	**8**	0b**1000**	0x**8**	*eight*
𝍸 ||||	ix	**9**	0b**1001**	0x**9**	*nine*
𝍸 𝍸	x	**10**	0b**1010**	0x**A**	*ten*
𝍸 𝍸 |	xi	**11**	0b**1011**	0x**B**	*eleven*
𝍸 𝍸 ||	xii	**12**	0b**1100**	0x**C**	*twelve*
𝍸 𝍸 |||	xiii	**13**	0b**1101**	0x**D**	*thirteen*
𝍸 𝍸 ||||	xiv	**14**	0b**1110**	0x**E**	*fourteen*
𝍸 𝍸 𝍸	xv	**15**	0b**1111**	0x**F**	*fifteen*

Bases **2**, **10**, and **16**

Binary		Decimal		Hexadecimal	
0000	**1000**	**0**	**8**	**0**	**8**
0001	**1001**	**1**	**9**	**1**	**9**
0010	**1010**	**2**	**10**	**2**	**A**
0011	**1011**	**3**	**11**	**3**	**B**
0100	**1100**	**4**	**12**	**4**	**C**
0101	**1101**	**5**	**13**	**5**	**D**
0110	**1110**	**6**	**14**	**6**	**E**
0111	**1111**	**7**	**15**	**7**	**F**

Radix Numeral Systems

R is the ***radix*** (or ***base***)

R in **N** and R > 0

A is a set of R ***numerals***

A represents {0, 1, . . . R–1}

Theorem: If **n** is in **N** then

There is a *k* in **N** and a_k, a_{k-1}, . . . a_0 in **A** such that

$$\mathbf{n} = \mathrm{a}_k \mathrm{a}_{k-1} \ldots \mathrm{a}_{0 \ \text{base R}} = \sum_{i=0}^{k} \mathrm{a}_i \mathrm{R}^i$$

$a_k = 0 \Rightarrow n=0$

and the representation is unique

Four Hundred and Thirty Seven

110110101_2	437_{10}	$1B5_{16}$
$1 \cdot 2^8 = 256$ $+\ 1 \cdot 2^7 = 128$ $+\ 0 \cdot 2^6 = 0$ $+\ 1 \cdot 2^5 = 32$ $+\ 1 \cdot 2^4 = 16$ $+\ 0 \cdot 2^3 = 0$ $+\ 1 \cdot 2^2 = 4$ $+\ 0 \cdot 2^1 = 0$ $+\ 1 \cdot 2^0 = 1$	$4 \cdot 10^2 = 400$ $+\ 3 \cdot 10^1 = 30$ $+\ 7 \cdot 10^0 = 7$	$1 \cdot 16^2 = 256$ $+\ 11 \cdot 16^1 = 176$ $+\ 5 \cdot 16^0 = 5$

Conversion to Base R

Repeated division

```
n₀ = n
while nᵢ ≠ 0
   aᵢ   = nᵢ rem R
   nᵢ₊₁ = nᵢ div R
```

$$\mathbf{n} = \sum_{i=0}^{k} a_i R^i$$

R = 10 $\qquad$ **n** = 437

$a_0 = 7, \quad n_1 = 43$

$a_1 = 3, \quad n_2 = 4$

$a_2 = 4, \quad n_2 = 0$

$437 = 4 \cdot 10^2 + 3 \cdot 10^1 + 7 \cdot 10^0$

Base **2** $\leftrightarrow$ Base **10**

From base **2** $\rightarrow$ base **10**

Add the power of 2 corresponding to each 1

Example: $01100100_2 = 2^6 + 2^5 + 2^2 = 64 + 32 + 4 = 100_{10}$

From base **10** $\rightarrow$ base **2**

Express number as sum of distinct powers of 2

$209_{10} = 128 + 64 + 16 + 1 = 1\cdot 2^7 + 1\cdot 2^6 + 1\cdot 2^4 + 1\cdot 2^0$

Add zero times the missing powers of 2

$209_{10} = 1\cdot 2^7 + 1\cdot 2^6 + 0\cdot 2^5 + 1\cdot 2^4 + 0\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0$

Write coefficients from highest to lowest power of 2

$209_{10} = 11010001_2$

Remainders of repeated division by 2

Powers of 2

2^0	1	2^8	256
2^1	2	2^9	512
2^2	4	2^{10}	1024
2^3	8	2^{11}	2048
2^4	16	2^{12}	4096
2^5	32	2^{13}	8192
2^6	64	2^{14}	16384
2^7	128	2^{15}	32968

Base **16** $\leftrightarrow$ Base **2**

From base 16 to base 2

Replace each hex digit with its 4-bit binary equivalent

$6E30AC58_{16} = 01101110001100001010110001011000_2$

From base 2 to base 16

Pad left with 0 until length is multiple of 4

$1100100100111011_2 = 0001100100100111011_2$

Replace consecutive sequences of 4 bits with hex digit

$0001100100100111011_2 = 1927B_{16}$

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Binary Natural Numbers

...00001111	15
...00001110	14
...00001101	13
...00001100	12
...00001011	11
...00001010	10
...00001001	09
...00001000	08
...00000111	07
...00000110	06
...00000101	05
...00000100	04
...00000011	03
...00000010	02
...00000001	01
...00000000	00

0, 1, 2, …

Closed under addition

Not under subtraction

A – B undefined iff B > A

HW 3: Conversion Between Bases

Convert the following values to the indicated base:

1011011101111110_2	$\rightarrow$	base 16
$4C1F91_{16}$	$\rightarrow$	base 2
100_{10}	$\rightarrow$	base 2
10001010_2	$\rightarrow$	base 10
$1F3_{16}$	$\rightarrow$	base 10
1055_{10}	$\rightarrow$	base 16