Signed Magnitude Notation

Wednesday, April 7, 2021 1:54 PM

Unsigned Binary - what you worked with before (ex 0010 as 2)

Signed Magnitude Notation - leftmost bit represents the sign (+/−); other bits represent the magnitude

Ex: Represent +5 using signed magnitude notation (using 4-bits)

sign	Magnitude		
0	1	0	1

↑

+ is 0

Ex: Represent −5 using s.m.n.

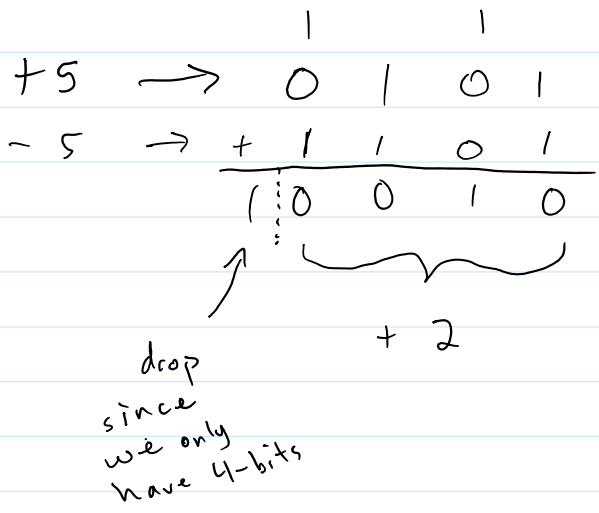
sign	Magnitude		
1	1	0	1

−5 is 1101

Ex: Represent 0 using signed magnitude notation.

sign	magnitude			
0	0	0	0	← +0 } two zeros; not ideal
1	0	0	0	← −0 }

Ex: Add −5 and +5 using S.M.N.



$\Rightarrow 5 + (-5) = 2$

Not good for simple arithmetic

One's Complement Notation

Wednesday, April 7, 2021 2:19 PM

Def: The complement of a binary number is formed by reversing every bit.

Ex: Complement the binary value 0100.

Sol: 1011

Ex: Represent +5 using one's complement notation.

Sol: 0101

↑ sign

↳ magnitude

Ex: Represent −5 using 0.C.N.

+5 −5

Sol: 0101 → (complement) → 1010

↑ sign ↑ magnitude

Ex: Represent +2 and −2 using 0.N.C

+2 −2

Sol: 0010 → (complement) → 1101

Ex: Represent 0.

Sol: 0000 +0

1111 −0

} still have two zeros

Ex: Add +5 and −5.

Sol:

$$\begin{array}{rr} +5 & 0\,1\,0\,1 \\ -5 & +\;1\,0\,1\,0 \\ \hline & 1\,1\,1\,1 \end{array} \longrightarrow -0_{10}$$

Try this: Add +5 and −2

Sol: +5 is 0101 +2 is 0010

−2 is 1101

$$\begin{array}{rr} & 1\;\;1 \\ +5 \longrightarrow & 0\,1\,0\,1 \\ +\;-2 \longrightarrow & +\;1\,1\,0\,1 \\ \hline & 1\,0\,0\,1\,0 \end{array}$$

drop ↑

1 0010 → sign mag } +2

Two's Complement Notation

Wednesday, April 7, 2021 2:35 PM

Positive values are the same as unsigned binary.

To negate values:

Step 1: Start with given value in binary

Step 2: Complement every bit.

Step 3: Add 1

Ex: Write −5 using two's complement

1.) +5 is 0101

2.) Complement 1010

3.) Add 1

$$\begin{array}{r} 1010 \\ +\ 0001 \\ \hline 1011 \end{array}$$

1011 is −5 using T.C.

Try this: Write −2 using two's complement

Sol: 1.) +2 is 0010

2.) Complement 1101

3.) Add 1 1110

$$\begin{array}{r} 1101 \\ +\ 0001 \end{array}$$

Try this: Convert −> (1001) to +7

Sol: 1.) -7 is 1001

2.) Complement 0110

3.) Add 1 0111 $\longrightarrow$ $+7$ is 0111

Ex: Find -0 from $+0$.

Sol: 1.) $+0$ is 0000

2.) complement 1111

3.) Add 1 0000 $\Leftarrow$

$$\begin{array}{r} {\scriptstyle 1\,1\,1} \\ 1111 \\ +\ 0001 \\ \hline 10000 \end{array}$$

drop

We only have 1 zero!!

Ex: Add $+5$ and -5

Sol: $+5$ is 0101

-5 is 1011 $\Rightarrow$

$$\begin{array}{r} {\scriptstyle 1\,1\,1} \\ 0101 \\ +\ 1011 \\ \hline 10000 \end{array}$$

drop 0_{10}

Ex: Add $+5$ and -2

Sol: $+5$ is 0101

-2 is 1110 $\Rightarrow$

$$\begin{array}{r} {\scriptstyle 1} \\ 0101 \\ +\ 1110 \\ \hline 10011 \end{array}$$

drop +3

Arithmetic works easily!!

We've worked with 4-bits of storage:

Min value is $-2^{4-1} = -2^3 = -8$

Max value is $2^{4-1} - 1 = 2^3 - 1 = 8 - 1 = 7$

In general with n-bits.

Min value is -2^{n-1}

Max value is $2^{n-1} - 1$

Overflow

Wednesday, April 7, 2021 3:00 PM

Def: Overflow – occurs when the value that is to be stored is outside the range of permissible values.

Handle by adding more bits or just detecting and reporting

Detecting Overflow:

Ex: Add +4 and +5

Sol:

```
      1
+5 is  0 1 0 1
+4 is +0 1 0 0
      1 0 0 1   → −7
```

Ex: Add −4 and −5

```
−4 is  1 1 0 0
−5 is +1 0 1 1
      1 0 1 1 1
        ↑ ~~~
      drop +7
```

sign is opposite the two original signs.

ASCII

Monday, April 12, 2021 1:34 PM

American Standard Code for Information Interchange

Character group	Range	
	Decimal	Hexadecimal
Control characters	0 – 31	0 – 1F
Punctuation	32 – 47	20 – 2F
Digits	48 – 57	30 – 39
More punctuation	58 – 64	3A – 40
Uppercase letters	65 – 90	41 – 5A
More punctuation	91 – 96	5B – 60
Lowercase letters	97 – 122	61 – 7A
More punctuation	123 – 126	7B – 7E
One final control character	127	7F

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	[space]	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	I	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

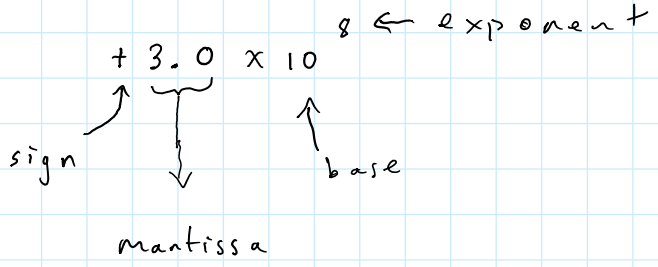
$A = 41_{16} = 01000001_2$

$a = 61_{16} = 01100001_2$

Floating Point Notation

Monday, April 12, 2021 1:34 PM

In base 10 we have exponential notation:



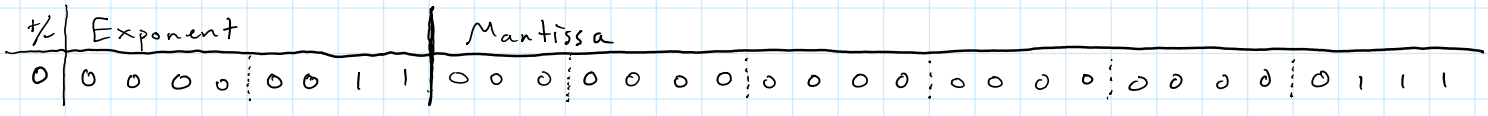
In floating Point notation we have 3 parts: (32-bits)

1.) sign for the mantissa
- 0 for positive
- 1 for negative

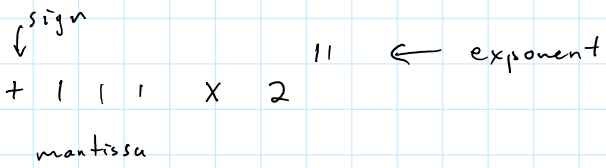
2.) Exponent
- 8-bit two's complement notation
- Base 2

3.) Mantissa
- Unsigned binary number
- 23 remaining bits

Ex: The number 56.0 in 32-bit floating point notation.



In exponential form



Convert to base 10 $7 \times 2^3 = 7(8) = 56.0$

In unsigned binary 1 1 1 0 0 0

In unsigned binary $1\,1\,1\,0\,0\,0$

(mantissa: $1\,1\,1$; 3 zeros from exp.)

Ex: Convert the 32-bit floating point number to base 10.

+/−	Exponent	Mantissa
1	1 1 1 1 1 1 1 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1

1.) Write the number in the binary exponential form

Sign is (−)

Mantissa is $11_2 \rightarrow 3_{10}$

Exponent is 11111110_2

-11×2^{11111110}

2.) Convert mantissa and exponent to base 10

For the mantissa:

$2^1 + 2^0 = 2 + 1 = 3$

For exponent (two's complement, 8-bit)

Given: 1 1 1 1 1 1 1 0

Complement: 0 0 0 0 0 0 0 1

Add 1: 0 0 0 0 0 0 1 0 → 2

exponent is -2

⇒ We have:

-3×2^{-2}

3.) Evaluate:

$-3 \times 2^{-2} = -3\left(\frac{1}{2^2}\right) = -3\left(\frac{1}{4}\right) = \frac{-3}{4} = \boxed{-0.75}$

Try this: Convert to base 10.

+/-	Exponent	Mantissa
1	1 1 1 1 1 1 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0

1.) Exponential form (binary)

$$-110 \times 2^{11111101}$$

2.) Conversion to base 10 for mantissa and exp.

Mantissa:

1 1 0

2^2 2^1 2^0

4 + 2 = 6

Exponent:

Given:	1	1	1	1	1	1	0	1
Comp:	0	0	0	0	0	0	1	0
Add 1:	0	0	0	0	0	0	1	1

→ 3

⇒ exp is −3

3.) Eval:

$$-6 \times 2^{-3}$$

$$-6 \times \left(\frac{1}{2^3}\right) = -\frac{6}{8} = -\frac{3}{4} = \boxed{-0.75}$$

Ex: Write +5.5625 in floating point notation. (32-bit)

Sol: 1.) split into 3 parts (sign, whole, fractional)

Sign:	Whole (left of .)	Fractional Part:
(+)	5	0.5625

2.) Convert each part to binary

Sign: 0

Whole:

5 ÷ 2 = 2 r 1

2 ÷ 2 = 1 r 0

→ 101

Fractional Part:

0.5625 × 2 = 1.125 = 1 + 0.125

0.125 × 2 = 0.25 = 0 + 0.25

0

5 ÷ 2 = 2 r 1

2 ÷ 2 = 1 r 0

1 ÷ 2 = 0 r 1 ↑

0.125 × 2 = 0.25 = 0 + 0.25

0.25 × 2 = 0.5 = 0 + 0.5

0.5 × 2 = 1.0 = 1 + 0

read down

1001

3.) Combine whole and fraction

101.1001

4.) Shift decimal for the exponent

101.1001

shifts 4 times to the right.

⇒ exponent is −4

⇒ mantissa is 1011001

In 8-bit TC:

Positive 4 is 00000100

Complement 11111011

Add 1 11111100 exponent

5.) Combine it all

±	Exponent	Mantissa
0	1111 1100	0000 0000 0000 0000 0101 1001

Try this: Convert −16.125 to floating point notation (32-bit)

Sol: 1.) Split it

Sign	Whole	Fractional
(−)	16	0.125

Sign	Whole	Fractional
(-)	16	0.125

2.) Convert each part to binary

Sign
1

Whole

16 ÷ 2 = 8 , 0
8 ÷ 2 = 4 , 0
4 ÷ 2 = 2 , 0
2 ÷ 2 = 1 , 0
1 ÷ 2 = 0 , 1

→ 1 0000

Fractional:

0.125 x 2 = 0.25
0.25 x 2 = 0.5
0.5 x 2 = 1.0

0 0 1

3.) Recombine whole and fraction

1 0 0 0 0 . 0 0 1

4.) Shift decimal for mantissa and exponent

Mantissa:
1 0000001

Exponent is -3:

Positive 3: 0 0 0 0 0 0 1 1

Complement: 1 1 1 1 1 1 0 0

Add 1: 1 1 1 1 1 1 0 1

5.) Combine it all:

S	Exponent	Mantissa
1	1 1 1 1 1 1 0 1	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1

Decoders

Wednesday, April 14, 2021 1:55 PM

Def: A decoder takes in a number (unsigned binary) and generates a 1 (high) on the output line that corresponds to the input number.

n input lines $\Rightarrow$ 2^n output lines

Ex: 2 to 4 decoder:

If the input is 00_2, then the output line 0 is high.

Input is 01_2 $\Rightarrow$ output line 1 is high.

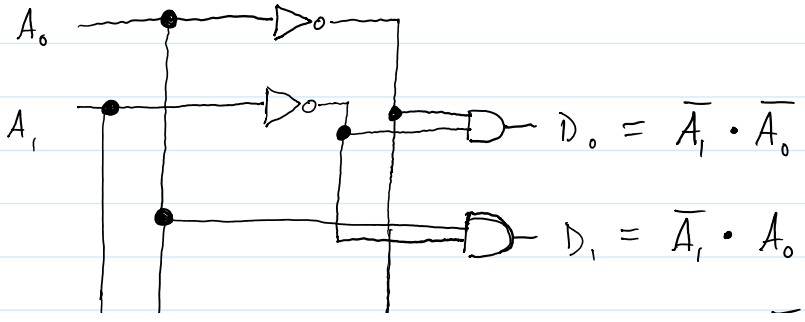
Input is 10_2 $\Rightarrow$ output line 2 is high

Input is 11_2 $\Rightarrow$ output line 3 is high

Truth table for 2 to 4 decoder:

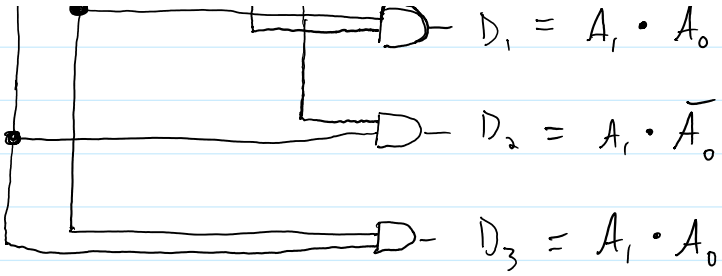
A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Circuit w/ Boolean Algebra:

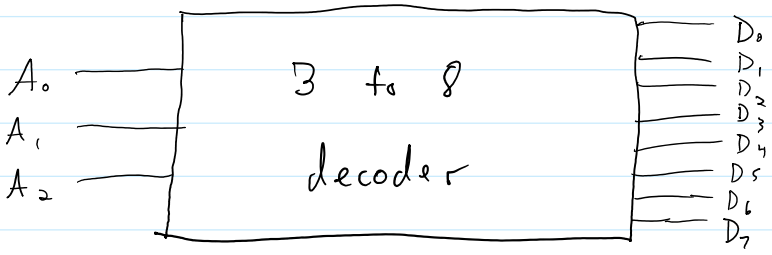


$\cdot$ is and

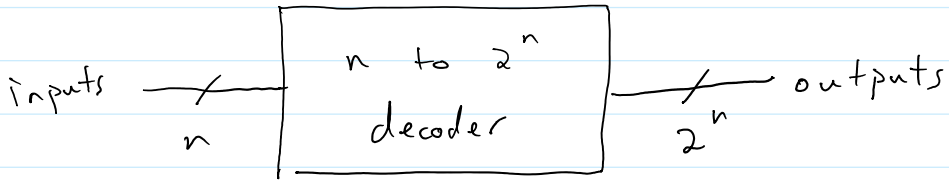
$\overline{}$ is not



Expressing a 3 to 8 decoder as a black box circuit:



n to 2^n decoder:



Encoders

Wednesday, April 14, 2021 1:55 PM

Def: An encoder has 2^n input lines, with only 1 being high at a time, producing an n-bit unsigned binary number.

Truth table (with valid input lines):

D_3	D_2	D_1	D_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

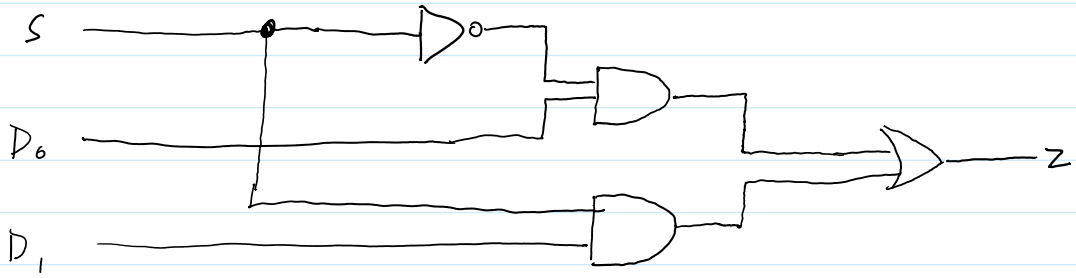
See www.thescienceofcomputing.com/circuits for a visual

Multiplexers

Wednesday, April 14, 2021 1:55 PM

Def: Multiplexers transfer multiple input data lines to a single output by using selectors.

Circuit: Two-Input MUX (2 input data lines, 1 selector)
General: 2^n data lines, n selector)

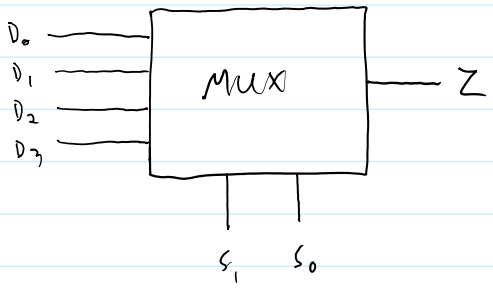


Boolean Algebra: $Z = D_0 \cdot \bar{S} + D_1 \cdot S$

Truth Table:

S	D_1	D_0	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Ex: 4 Input MUX (as Black Box)

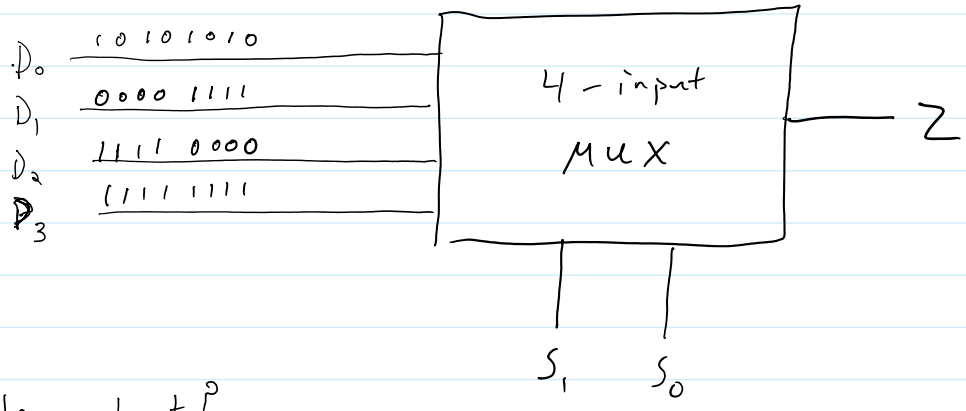


$$Z = D_0 \cdot \overline{S_1} \cdot \overline{S_0} + D_1 \cdot \overline{S_1} \cdot S_0 + D_2 \cdot S_1 \cdot \overline{S_0} + D_3 \cdot S_1 \cdot S_0$$

If $S = 10$ ($S_1 = 1$ and $S_0 = 0$), then D_2 is put through

If $S = 11$ ($S_1 = 1$ and $S_0 = 1$), the D_3 is put through

Ex: Consider the following MUX. Binary values represent datastreams



What is the output?

a.) If $S = 00$ (S_1, S_0):

$Z = 10101010$ ← from D_0

b.) If $S = 01$

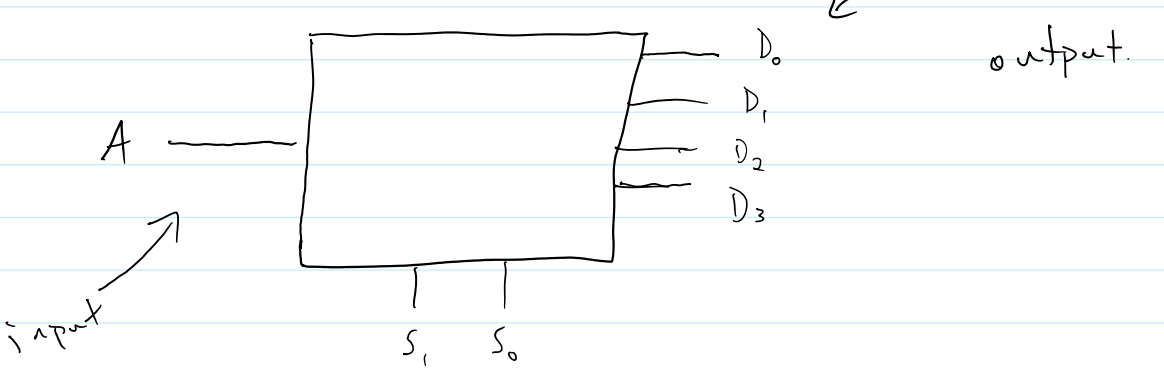
$Z = 0000\ 1111$ from D_1

Demultiplexers

Wednesday, April 14, 2021 1:56 PM

Def: Demultiplexers have a single input line, n selector inputs, and 2^n output data lines.

Black Box for 4-output DEMUX:



Flip Flops

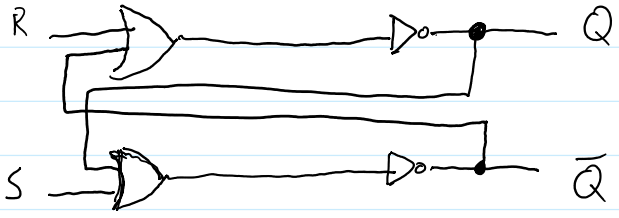
Wednesday, April 14, 2021 1:57 PM

Def: Sequential Circuits – circuits that "have memory". An output (or several) is fed back as input.

Def: Flip flop – a device having 2 stable states; used to store a 0 or a 1.

R-S flip flop:

- R is for Reset
- S is for Set
- Q is the state
- Q holds the state until a signal changes it.



Truth table:

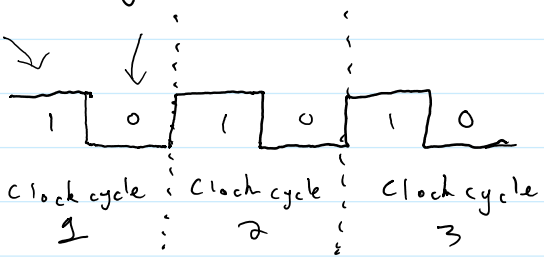
S	R	Q	$\overline{Q}$
0	0	no change	
0	1	0	1
1	0	1	0
1	1	Not allowed bc	$Q \neq \overline{Q}$

Clocked R-S flip flop:

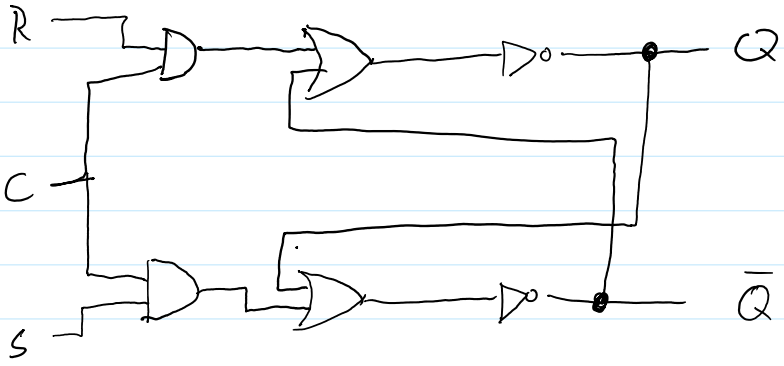
Def: Clock – a device that cycles between high and low states

Def: Clock cycle – time interval lasting from high through low

Def: Clock cycle time ... lastly from ... through low



Circuit: Clocked R-S Flip flop



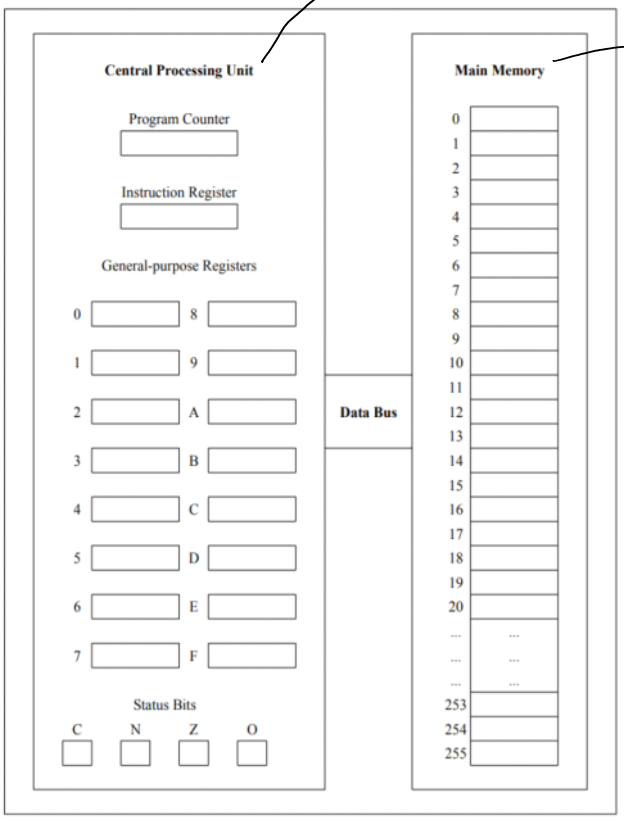
The clock C ensures our memory, Q, cannot be changed during a "read" phase (when C = 0)

A Simple Computer - Part 1

Friday, April 16, 2021 1:33 PM

Register - like memory, stores data within the CPU

CPU - performs arithmetic/logic; the brains



RAM - stores program and the data on which it acts
256 - slots, each storing a word of data. (A word is the base unit of data for the machine)

Program Counter - special purpose register for the next instruction

Instruction Register - for the current instruction.

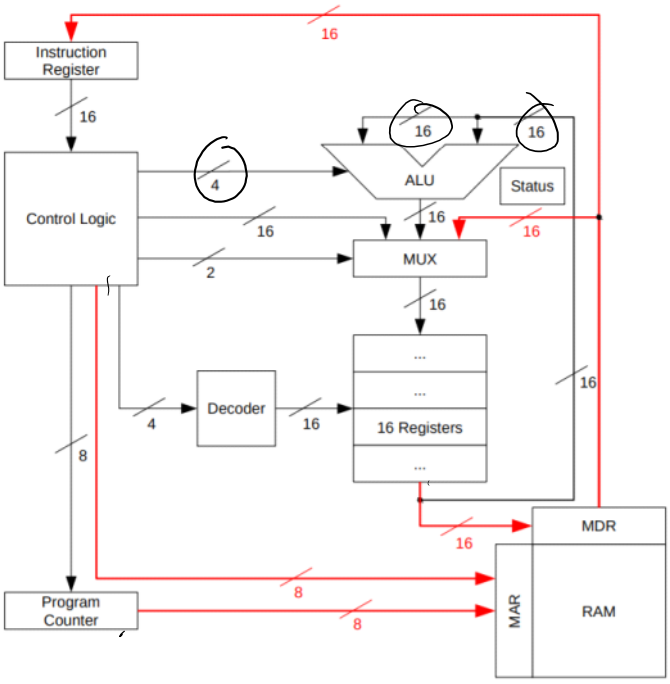
General purpose Registers - manipulated on by arithmetic and logic units of the CPU

Status Bits: registers that contain info about our most recent computations.

C = carry
N = Negative
Z = Zero result
O = overflow

A Simple Computer - Part 2

Friday, April 16, 2021 1:36 PM



New Parts:

MDR - memory data register holds data to be read by the CPU or written to memory

MAR - memory address register; holds address of memory location to be accessed.

Control Logic/Unit - implements the instruction cycle. The "traffic cop".

ALU - Arithmetic Logic Unit; does math and logic operations

Instruction Cycle (5 tasks):

1.) Fetch - get the next instruction from memory

2.) Increment - increment the program counter by pointing to the next instruction

3.) Decode - decode the current instruction; identify the op-code and operands

op-code ↑ defines the operation

4.) Execute - performs the instruction
Routes operands to appropriate hardware

Results are computed and routed to the appropriate destination.

5.) Return to step 1