CHAPTER 1 - Intro

- Algorithm -- step-by-step procedure, guaranteed to finish
- Program -- express the algorithm using a computer language
- Instruction Set Architecture (ISA) -- specifies the set of instructions the computer can perform
- Microarchitecture -- detailed organization of a processor implementation
- Logic Circuits -- combine basic operations to realize microarchitecture

CHAPTER 2 – Bits, Data Types, and Operations

Unsigned Integers

$$10101 = 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 21$$

Signed Integers

Sign-Magnitude – Set MSB to 1.

One's Compliment – Flip every bit.

Two's Compliment – Copy bits up to and including the first 1. Flip the remaining left bits.

Converting 2C to Decimal

1. If leading bit is one, take 2C to get a positive number.
2. Add powers of 2 for each position.
3. If original number was negative add a minus sign.

$$0b01101000 = 2^6 + 2^5 + 2^3 = 64 + 32 + 8 = 104$$

$$0b11100110 = 0b00011010 = 2^4 + 2^3 + 2^1 = 16 + 8 + 2 = 26 = -26$$

$$0b0000.111 = 1*2^{-1} + 1*2^{-2} + 1*2^{-3} = 0.5 + 0.25 + 0.125 = 0.875$$

Converting Decimal to 2C Binary

1. Find magnitude of decimal number.
2. Divide by two – remainder is MSB
3. Keep dividing by two until answer is zero, write remainders from right to left.
4. Prepend a zero as the MSB. If original number was negative, take the 2C.

104/2	=	52 r0	bit 0	6/2	=	3 r0	bit 4	104 0b01 101000
52/2	=	26 r0	bit 1	3/2	=	1 r1	bit 5	
26/2	=	13 r0	bit 2	1/2	=	0 r1	bit 6	

13/2 = 6 r1 bit 3		

2C Addition

Same as binary addition. Ignore carry out.

2C Subtraction

Negate the 2nd number then add. Ignore carry out.

Logical Operations

- AND – Useful for clearing bits. “Bitwise multiply”
- OR – Useful for setting bits. “Bitwise add without carry”
- NOT – One argument. Flips every bit.

Decimal -> Binary -> Hex -> Octal

Dec	Binary	Hex	Octal		Dec	Binary	Hex	Octal
0	0000	0	0		8	1000	8	10
1	0001	1	1		9	1001	9	11
2	0010	2	2		10	1010	A	12
3	0011	3	3		11	1011	B	13
4	0100	4	4		12	1100	C	14
5	0101	5	5		13	1101	D	15
6	0110	6	6		14	1110	E	16
7	0111	7	7		15	1111	F	17

Binary To Hex Conversion

From the LSB group the digits into groups of 4. Use the LUT.

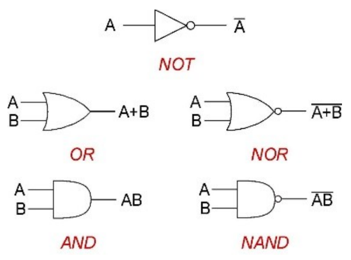
CHAPTER 3

Transistor Types

- CMOS – Complementary Metal Oxide Semiconductor
- n-Type – When Gate is pulled High, current flows from source to drain.
- p-Type – When Gate is pulled Low, current flows from source to drain.

Logic Gates

- Combinational Logic Circuit – stateless, output depends on the current inputs.
- Sequential Logic Circuit – holds state, output depends on sequence of inputs.
- Gates:
  - NOT – Inverter
  - NOR – Only true when both inputs are false.
  - OR – True when either input is true.
  - NAND – Always true unless both inputs are true.



  - AND – Only true when both inputs are true.
- NOT all of AND to make an OR.
- NOT the output of AND to make a NAND.
- NOT the output of OR to make a NOR.

<u>Implementing Truth Tables</u>

1. AND combinations that are 1 in the truth table.
2. OR the results of the AND gates.

<u>R-S Latch</u>

- R=S=1 – Hold value
- S=0,R=1 – set value to 1
- R=0,S=1 – set value to 0
- R=S=0 – this is bad!

CHAPTER 4 – Architectures

- Instruction Register (IR): contains the current instruction.
- Program Counter (PC): contains the address of the next instruction to be executed.
- Control Unit: Reads and instruction from memory, interprets it, and then generates control signals.
- Instruction Set Architecture – A computer's instructions and their formats.
- MAR: Memory Address Register
- MDR: Memory Data Register

LOAD:	STORE:
1. Write address to MAR 2. Send read signal. 3. Read the data from MDR.	1. Write the data to the MDR. 2. Write the address into the MAR. 3. Send a write signal to the memory.

<u>Instruction Processing Cycle "FDEFES"</u>

1. Fetch Instruction a. PC -> PMAR b. PMDR -> IR c. PC++ 2. Decode Instruction 3. Evaluate Address	4. Fetch Operands 5. Execute 6. Store Result

CHAPTER 5 – Assembly Language

- Symbol Table – all labels and corresponding addresses
- First pass – generate symbol table.
- Second pass – convert instructions to machine language using symbol table.
- Loader – copies executable image into memory.
- Linker – Resolves symbols between independent object files.

CHAPTER 6 – Atmega328p

ISA/Specifications

- Program Mem: 32k 16-bit
- Data Mem: 2k 8-bits
- Registers: 32x R0-R31 each 8 bits wide
- OPCODES:
  - *ALU* instructions: ADC, ADD, AND, NOP, CP, EOR, OR, MOV
  - *Immediate* instructions: CPI, ORI, ANDI, LDI
  - *Unary Logical* instructions: COM, NEG, ASR, LSR
  - *Load/Store* instructions: LDS, STS
  - *Branch* instructions: BRBS, BRBC (aka BRZS, BRZC, BRCS, BRCC)
  - Conditional branch based upon SREG bits (“Condition Codes”)
  - *Input/Output* instructions: IN, OUT
  - *Call/Jump* instructions: CALL, JMP
  - *Return* instructions: RET, RETI
  - *Stack* instructions: PUSH, POP
  - *Relative Jump* instructions: RCALL, RJMP

Status Register (SREG)

I	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
Interrupt	Transfer	Half Carry	Signed	2C Overflow	Negative	Zero	Carry

CHAPTER 7 – The Stack

- LIFO – Last In First Out
- Stack Pointer (SP): Points to the empty top of the stack.
- PUSH: adds an item to stack. SP--
- POP: remove item from stack. ++SP
- CALL/RCALL: Push PC to stack then vector to a label. SP = SP-2
- RET/RETI: Pop addr fro
- m Stack to PC. SP=SP+2