Add diagrams, pg 35
2's complement: add 1!

Instruction Execution Cycle:********
1.Fetch instruction 2.Increment IP 3. Decode 4.Fetch operands 5.Store result
CISC (Complex Instruction Set Computer)********
Peripheral Devices: External
non-volatile storage
convert human/machine readable forms
I/O Unit: Communcate between CPU, peripherals
Main Memory Unit: stores programs, volatile
CPU: ALU(integer unit, floating point),clock,register,control unit

Bus: parallel wires, xfer signal internal, control, address, data,
wider=faster, longer=slower, need arbitration
Register: local memory in CPU
Arithmetic/logic unit
Microproram/Micromemory: implemented in memory
better speed: cache, separate I/O unit, separate network unit

Registers:**********
Control: current state of machine
Status: status of operation
MAR: memory address register, holds address
MDR: memory data register, holds data MAR points to
IP: instruction pointer
IR: instruction register

Operand_1,Operand_2,Result:ALU registers
General: fast temp storage
Cache: fast temp storage copied from slower storage
instructions moved from secondary to main
data moved from main to CPU registers

****
bit = 1
byte = 8
word = 16
double word = 32
quadword = 64
double quadword = 128

fword = 48
qword = 64
tbyte = 80
real4 = 32 bit IEEE short real
real8 = 64 long real
real10 = 80 extended long real

****
kilobyte = 2^{10}
megabyte = 2^{20}

gigabyte = 2^{30}
VonNeumann architecture: program stored in memory, done by OS

modes of operation:********
protected: native, features available,programs given segments
virtual 8086: multiple reall address mode

real address mode: environment of early intels
system management mode: power management, security

intel ia-32: 4GB, byte addressable, little endian

```
.386
.model flat,stdcall
.stack 4096
ExitProcess proto,dwExitCode:dword
.data
	; declare variables here
.code
main proc
	; write your code here
	invoke ExitProcess,0
main endp
end main
```

General Notes:

sign stored in MSB
integer range of ascii codes:0-127
UTF-8 same as ascii
UTF-16 16 bits, used by windows
UTF-32 32 bits, space no concern, fixed width chars required
integer in bits: 2^{31}-1
AH(high); AL(low) 8bit, AX 16bit, EAX 32 bit
Hamming Code: xx0x000x0000, $\log_2 n+1$, 2^n

Flags:********
carry: unsigned overflow
overflow: signed overflow
zero: operation produced zero
sign: negative result
parity: even number 1's in LSB
auxiliary carry: carries out of position 3 in LSB

creating stack frame:
arguments pushed
subroutine return address pushed
EBP pushed
ebp equal esp
esp decremented
registers pushed

stack frame notes:
ESP points to last value added to stack
stack parameters: reference, value

stack frame: set aside for passed arguments, subroutine return address,
local vars, saved registers
argument passed by reference = offset of object

cmp results:********
dest<sour: ZF 0, CF 1
dest>sour: ZF 0, CF 0
dest=source: ZF 1, CF 0

single precision: sign:1 exp:8 significand: 23
double precision: sign:1 exp:11 significand:52
double extended precision: sign:1 exp:16 significand:63

****
CloseFile-close disk file
Clrscr-clear screen
CreateOutPutFile-new disk file
Crlf-endline
Delay-Pause for n millisecond
DempMem-mem block to console in hex
DumpRegs-regs, flags to console in hex
GetCommandTail-command line args into array
GetDateTime-date and time

GetMaxXY-#rows, columns in window
GetMSeconds-#milliseconds since midnight
GetTextColor-foreground,background text color
Gotoxy-move cursor
IsDigit-sets 0 if Al=digit,0-9
MsgBox-message box
MsgBoxAsk-msg box w/ yes/no
OpenInputFile-open disk file
ParseDecimal32-unsigned decimal to string
ParseInteger32-signed decimal to string
Random32-32 bit integer
Randomize-set random seed
RandomRange-random w/in range
ReadChar-reads char
ReadDec-read unsigned
ReadFromFile-input file into buffer
ReadHex-read hex int
ReadInt-32 bit signed decimal
ReadKey-char w/out waiting
ReadString-read string
SetTextColor-sets foreground,background color
Str_compare-compare strings
Str_copy-copy strings
Str_length-length/string
Str_trim-remove chars
Str_ucase-change to uppercase
WaitMsg-message,wait for key
WriteBin-write 32bit int in ascii
WriteBinB-write in byte,word,dword format
WriteChar-write char
WriteDec-unsighned,32bit
WriteHex-32bit hex
WriteHexB-write byte,word,doubleword int in hex
WriteInt-32bit signed
WriteStackFrame-current procedures stack frame, to console
WriteStackFrameName-cur procedure name,stack frame
WriteString-write string
WriteToFile-write byffer to output file
WriteWindowsMsg-string w/most recent windows error

****

CMP Flags:
JC-jump if carry
JNC-jump if not carry
JZ-jump if zero
JNZ-jump if not zero
JO-jump if overflow
JNO-not overflow
JS-signed
JNS-not signed
JP-parity
JNP-not parity

Unsigned Comparisons:
JA-above >
JNBE-not below or equal
JAE-above or equal >-
JNB-not below
JB-if below <
JNAE-not above or equal
JBE-below or equal
JNA-not above

Signed Comparison:
JG-greater >
JNLE-not less than or equal
JGE-greater than, equal >=
JNL-not less
JL-less <
JNGE-not greater than, equal
JLE-less than, equal <=
JNG-not greater

****

Passing Parameters:
1st pushed has biggest offset
Indirect-accessing through registers-reference arrays in proc
-mov esi,[EBP+8]-add esi,4
Indexed-array name w/distance to element list[edi], add edi, 4-gloabal array refs
Base indexed-starting address in one,offset another-
reference arrays in proc-mov edx,[ebp+8],mov ecx,20 mov[edx+ecx]

RISC-reducied instruction set computer
longer, but faster (executed in hardware)
only LOAD, STORE reference memory

offset: distance bytes from beginning of segment, OS adds address
ptr: access different size, al byte
ptr[myDouble+1]

2D arrays: row first

lodsb: [esi] into AL, if direction flag=0,inc esi
stosb: move AL to [edi], edi inc if dir flag=0

FPU-registers=ST(0)..ST(7)
FINIT: initialize FPU registers-reference
FSUB,FADD etc—FISUB,FIADD for internal
FDIVR-division, operands reversed
FLD-push ST(i) to ST(i+1)
FST-top of stack to mem-FSTP-pop top

macroname MACRO [param1,param2..]
ENDM
use LOCAL for labels
params=memory,register,literal

Distributive Law:
A+BC=(A+B)(A+C)
A(B+C)=AB+AC
Absorption Law:
A(A+B)=A
A+AB=A

memory types: RAM-static RAM, dynamic RAM, synchronous DRAM
ROM-programmable ROM, erasable ROM, electrically EPROM

****

Instruction parallelism: pipeline, cache
processor: multiprocessor, multicomputer
Amdahls Law-speedup: n/(1+(n-1)f)-T/speedup
maybe probs w/ decision, repetition,procedure structures
multiprocessor-hard build, easy programs
communicate w/circuits
multi computer-easy build, hard program
comms w/packets, links, switches
hybrid-cloud computing

Parallel Perfomance depends on:
hardware:
CPU, I/O speed, interconnection, scalability
latency, bandwidth scalability-more processors affects bandwidth

software:
parallelizabililty, language, OS, parallel system library

applications: networks, internet
make things faster (chess, AI, expert systems)