Open file

EAX = 5 = sys_open

EBX = src file = "src.txt"

ECX = access mode = 0 = rd, 1 = wr, 2 = rd wr

EDX = permission = 777o

Int 80h

return EAX = file descriptor

Close file

EAX = 6 = sys_close

EBX = file descriptor

ECX = none

EDX = none

Int 80h

Read file

EAX = 3 = sys_read

EBX = file descriptor

ECX = buffer

EDX = buffer size

Int 80h

return EAX = bytes read

Write file

EAX = 4 = sys_write

EBX = file descriptor

ECX = buffer

EDX = bytes read

In 80h

Create file

EAX = 8 = sys_creat

EBX = file path = "new.txt"

ECX = permission

EDX = none

Int 80h

return EAX = file descriptor

mov

movzx ; zero extend

movsx ; sign extend

xchg ; exchange mem, reg or two regs

[ ] dereference a label

String operations

scd compare AL/AX/EAX to EDI

sto store AL/Ax/EAX into EDI

lod store ESI into AL/AX/EAX

mov copy ESI to EDI

cmp compare ESI to EDI

Direction flag controls directions (inc = clear / dec = set)

rep a repeat prefix, ECX controls loop

Multiplication

Multiplier	Multiplicand	Product
r8 / m8	AL	AX
r16 / m16	AX	DX:AX
r32 / m32	EAX	EDX:EAX

Carry flag is set if upper half of product contains significant digits

Division

Divisor	Dividend	Quotient	Remainder
r8 / m8	AX	AL	AH
r16 / m16	DX:AX	AX	DX
r32 / m32	EDX:EAX	EAX	EDX

Sign Extension Operations

cbw (convert byte to word)

cwd (convert word to dword)

cdq (convert dword to qword)

ex.
```
mov al, -48
cbw          ; extend AL to AH
mov bl, 5
idiv bl      ; AL = -9, AH = -3
```

Caller saved Registers

EAX, ECX, EDX only if the info is needed

Callee saved Registers

EBX, ESI, EDI, EBP only if used

General use Registers

EAX, EBX, ECX, EDX

EAX, EDX are used for arithmetic operations

EAX is used to hold return value from procedures

ECX is used for loop control

EBX

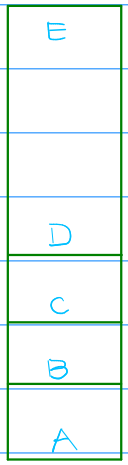
A Text - store machine code

B Data - store initialized global variables

C BSS - store uninitialized global variables

D Heap - dynamic memory allocation

E Stack - store local variables, function calls



Three instructions of CPU

Fetch - fetch instruction from register / memory

Decode - decode the instruction

Execute - execute the instruction

Command to assemble

nasm -f elf -g main.asm

ld -m elf_i386 main.o -o main.out

./main.out or gdb main.out

Shift Operations

SHL - shift all bits to left, lowest bit is filled with 0
highest bit is copied into carry flag

SAL - same as SHL

SHR - shift all bits to right, lowest bit is copied into carry flag
highest bit is filled with zero

SAR - mostly same as SHR but
highest bit (sign bit) is preserved

ROL - shift all bits to left, highest bit is copied into
carry flag and lowest bit

ROR - shift all bits to right, lowest bit is copied into
carry flag and highest bit

RCL - shift all bits to left, highest bit is copied into
CF and CF is copied into lowest bit

RCR - shif all bits to right, lowest bit is copied into
CF and CF is copied into highest bit

SHLD - shift all dst bits to left, highest bit is copied
into CF, highest src bit is copied into lowest dst bit

SHRD - shift all dst bits to right, lowest bit is copied
into CF, lowest src bit is copied into highest dst bit

Flags

zero flag - set if dst equal zero
sign flag - set if dst is negative
carry flag - set if unsigned value is out of range
overflow flag - set if signed value is out of range
direction flag - control inc/dec of ESI / EDI

Jump based on flag

jb, jc - jump if CF is set
je, jz - jump if ZF is set
js - jump if SF is set
jo - jump if OF is set
jp - jump if PF is set

Convert decimal to binary and hex

16 bits

301 0000 0001 0010 1101

0 1 2 d

301
−256
45
32
13
8
5
4
1
1

−450 0000 0001 1100 0010

2's 1111 1110 0011 1101

1

1111 1110 0011 1110

f e 3 e

450
−256
194
128
66
64
2
2

Add

```
0101 1110 0011 1111      24127
0010 0101 0101 1001       9561
---------------------------------
1000 0011 1001 1000      33688
```

Signed

```
1011 1010 0011 1101     -17859
1100 1010 1100 0111     -13625
---------------------------------
1000 0101 0000 0100     -31484
```

Subtract

0101 1110 0011 1111 — 24127
0010 0101 0101 1001 — 9561
0011 1000 1110 0110 — 14566

0101 1110 0011 1111
0010 0101 0101 1001 → 0010 0101 0101 1001 2'c
1101 1010 1010 0111
1
1101 1010 1010 0111

0101 1110 0011 1111
1101 1010 1010 0111
0011 1000 1110 0110

---

signed 1100 0110 0001 1111
1011 0010 1111 0011 → 1011 0010 1111 0011
0100 1101 0000 1100 2'c
1
0100 1101 0000 1101
-14817

-14317
-(-19725)
4908

1100 0110 0001 1111
0100 1101 0000 1101
0001 0011 0010 1100

32768 16384 8192 4096 2048 1024 512 256 128 64 32 16 8 4 2 1

---

signed 1110 1110 0111 1101
1011 0110 1100 1000 → 1011 0110 1100 1000 2'c
0100 1001 0011 0111
1
0100 1001 0011 1000

1110 1110 0111 1101
0100 1001 0011 1000
0011 0111 1011 0101
