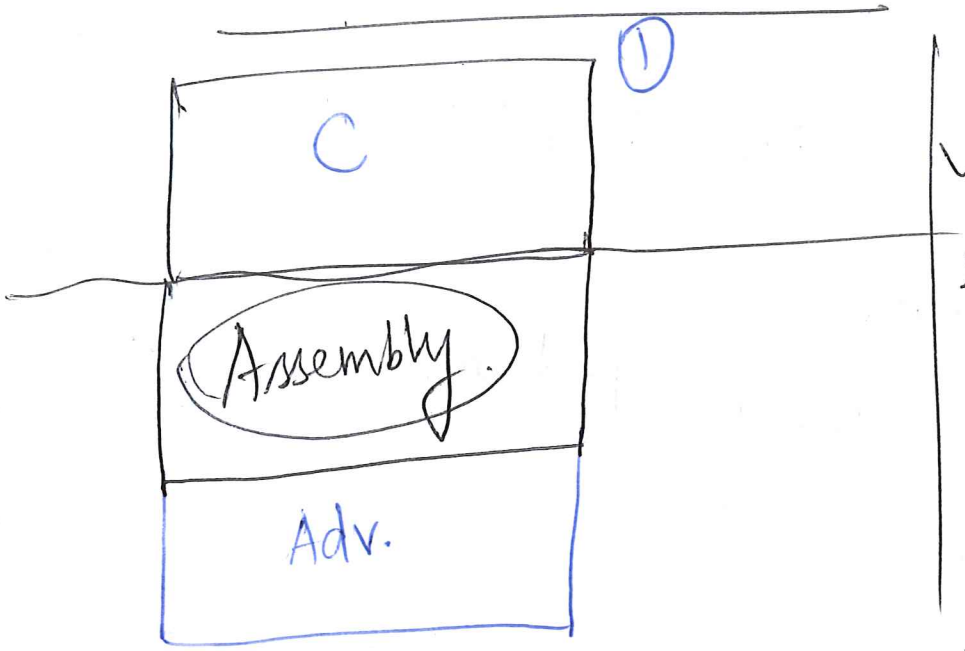
CS 354 - Lecture 6



1. Registers
2. Instructions
   - data transfer
   - arithmetic
3. Memory

C program → compiler → machine code (1010101 . . . 10)

C → Assembly → object file (binary)

Assembly: human read (mostly).

object file: NOT human-read.

more human readable

sum = x + y;

```
addl op1, op2
```

Assembly

Intel | ARM | SPARC

X 86 Assembly

8086, 80186, 80286, Pentium,

X 86

AT&T (Unix)

add A, B (source, dest)

B ← A+B

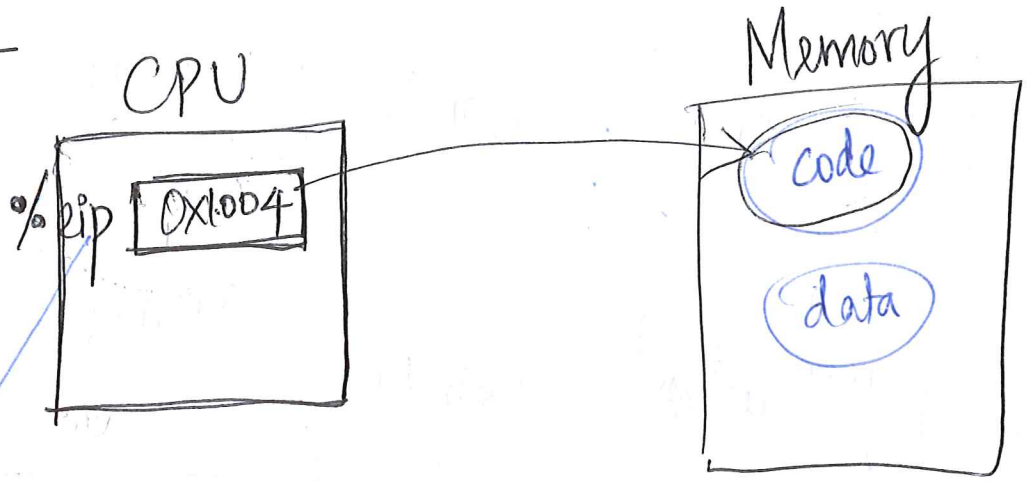
Intel

MS-DOS, Windows.

add A, B (dest, src.)

A ← A+B

Review



Program Counter
OR
Instruction Pointer.

Process

fetch
decode
execute

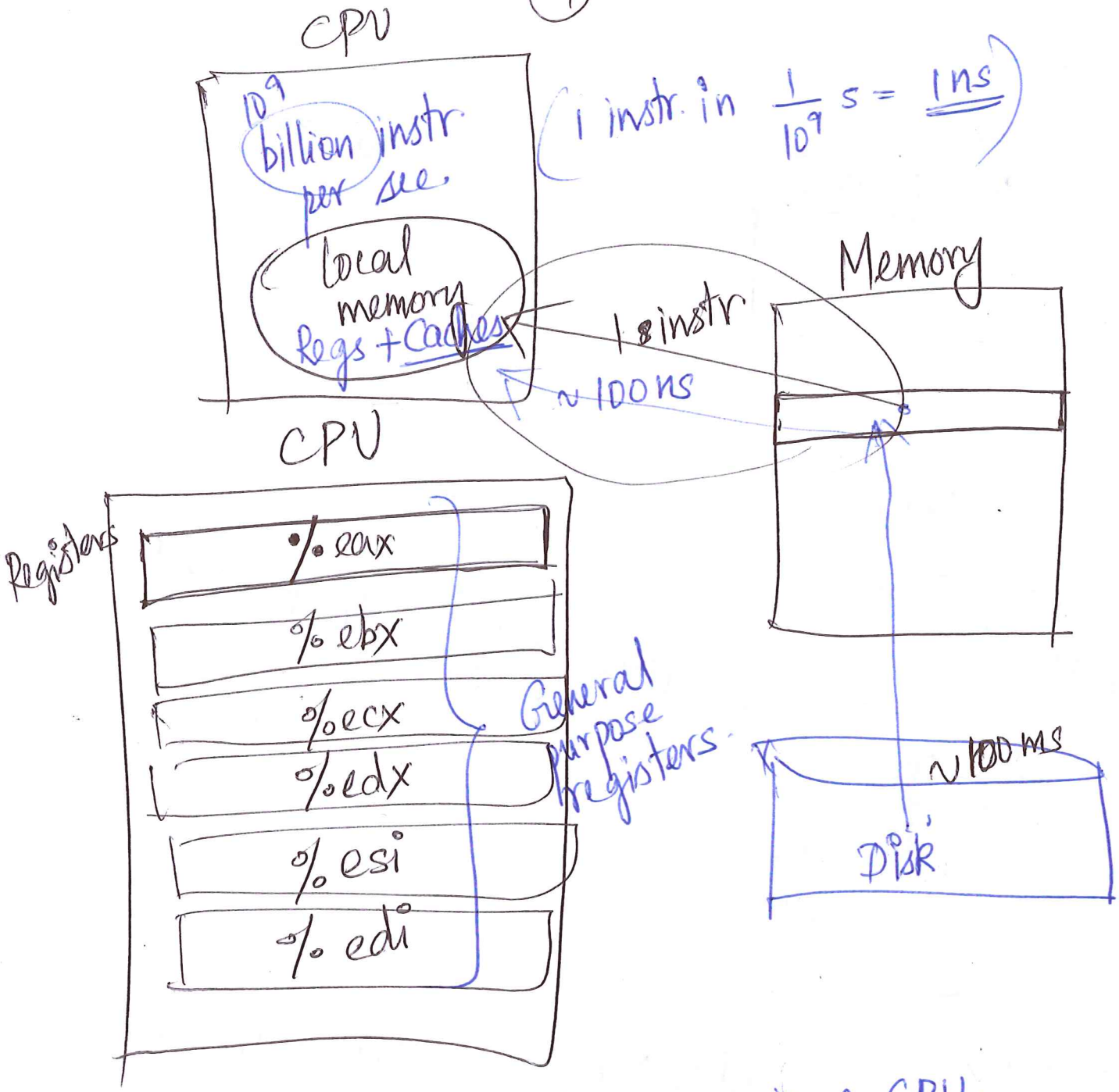
next time should eip point to next instr.

sum = x + y

add A, B

1011 0011 1111

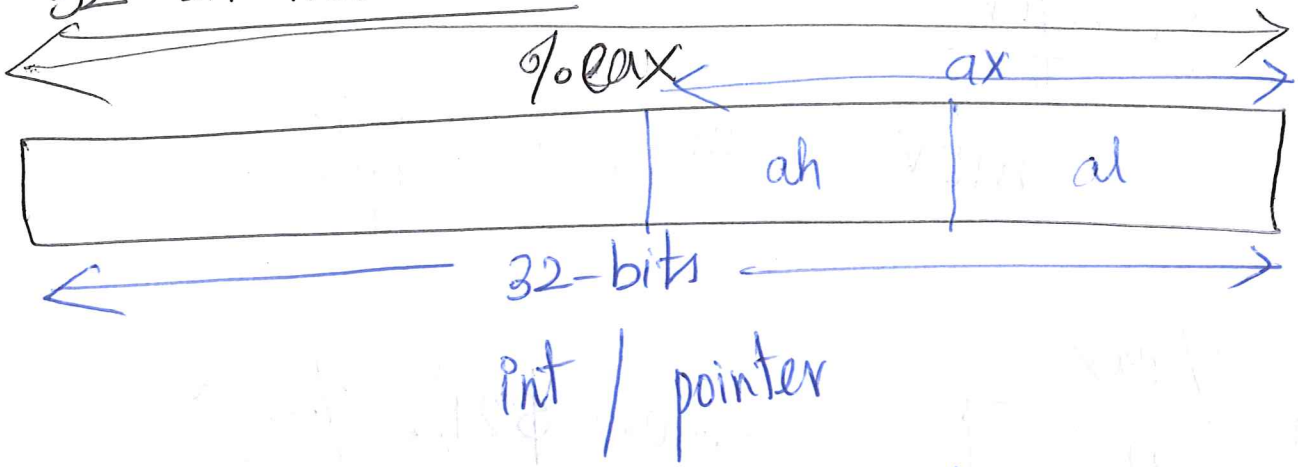
assembler

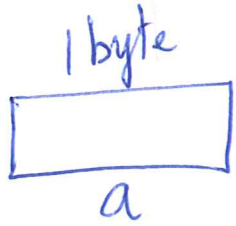


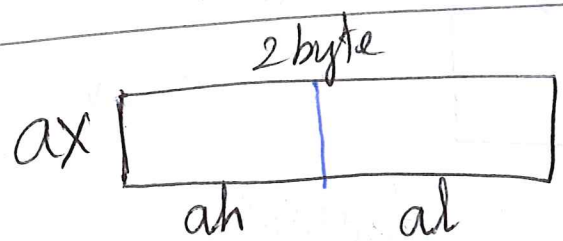
Registers

- small regions of memory in a CPU.
- each have a name
- read / write to them.

32-bit machines







Instructions

instruction set

- data transfer
- arithmetic
- memory
- control flow (if, functions, while)

Data transfer

mov source operand, dest. operand

%eax

21

mov $21, %eax

can't see in x86.

immediate value.

%ebx

21

mov %eax, %ebx

Variants

- movl → long word (4 bytes)
- movw → 2 bytes (movw $12, %ax)
- movb $10, %al
  - b → byte

movl $21, %eax.

Arithmetic instructions (7)

addl source, dest

dest ← source + dest

subl src, dest

dest ← dest − src

imull src, dest

int

long word (4 bytes)

dst ← dest * src

alt: imull aux, src, dest

dest ← aux * src

idivl operand

↓

divisor

9 / 2 → Q = 4, R = 1

dividend ↓ divisor

quotient → %eax

remainder → %edx

	unsigned	signed (8)
0 0 0	0	0
0 0 1	1	+1
0 1 0	2	+2
0 1 1	3	+3
1 0 0	4	−4
1 0 1	5	−3
1 1 0	6	−2
1 1 1	7	−1

end of class

4 2 1

1 0 0

↓

−4 + 0 + 0 = (−4)

(1) 0 1

4 2 1

↓

−4 + 0 + 1 = (−3)

Unsigned

5 + 3

11

101

011

(1) 000

carry

1 − 2

~~212~~

001

010

(111) (7)

3 − 2

011

− 010

0001

Signed

1 − 2 ⟹ 1 + (−2)

1 → 001

+ (−2) → 110

111

~~10~~

~~20~~

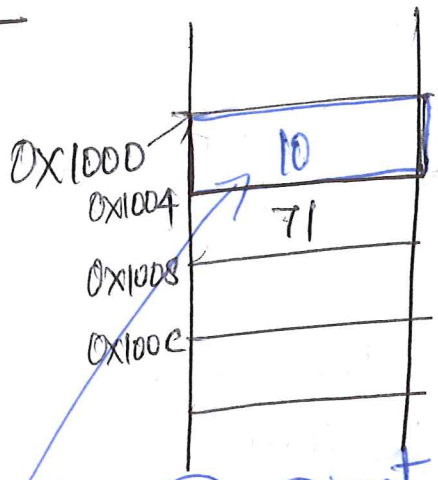
−4 − 1 ⟹ (3)

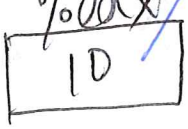
−4 → 100

+ (−1) → 111

~~(1)~~ 011

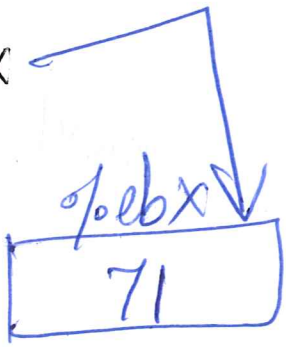
Memory





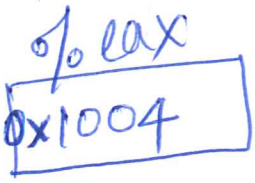
1 Direct access (Absolute)

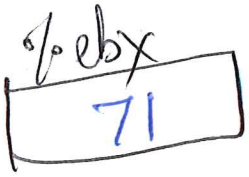
```
movl %eax, 0x1000
movl 0x1004, %ebx
movl $0x1004, %ebx
```



2 Indirect memory access

```
movl (%eax), %ebx
```





③ movl 8(%eax, %ebx, 4), %ecx

Scaled Index

Imm R1 R2 scaling

contents of the memory location

$$addr = 8 + \%eax + (\text{contents of } \%ebx * 4)$$

movl Imm(R1, R2, S), %R3

$$Imm + R_1 + S * R_2 \Rightarrow \text{memory location.}$$

---

movl 8(%eax), %ecx

%eax
0x1000

0x1000
+ 0x 8
0x1008

0x1008	21

%ecx
21

movl (%eax, %ebx, 4), %eax

addr = %eax + 4 * %ebx

movl 8(, %ebx, 4), %ecx

addr = 8 + 4 * %ebx

---

movl _____, Register

%eax

movl (%eax, %ebx, 4), %ecx

base

index

int

Problem #1

Write assembly to:
- move value 1 into %eax
- add 10 to it and put result into %eax

movl $1, %eax
addl $10, %eax

Problem #2

Expression: 3 + 6 * 2
Use one register (%eax), and 3 instructions to compute this piece-by-piece

movl $6, %eax
imull $2, %eax
addl $3, %eax

%eax: 6 → 12

2x6 = 12

Problem #3

```
movl $0, %edx
movl $7, %eax
movl $3, %ebx
idivl %ebx
movl %eax, %ecx
movl $0, %edx
movl $9, %eax
movl $2, %ebx
idivl %ebx
movl %edx, %eax
addl %ecx, %eax
```

S D

%edx: 0 → 1 | %eax: 7 → 2 | 1 → 3

%ebx: 2 → 3

7/3 → Q = 2, R = 1

%ecx: 2 → 2

Write simple C expression that is equivalent to these instructions

D ← S+D

7/3 + 9%2

Q(2) R(1)

9/2 → Q = 4, R = 1

Problem #4 (from CSAPP 3.1)

Memory

Address	Value
0x100	0xFF
0x104	0xAB
0x108	0x13
0x10C	0x11

Registers

%eax	0x100
%ecx	0x1
%edx	0x3

movl %eax, R

0x100

R

movl 0x104, R

Value of:

%eax	0x100
0x104	0xAB
$0x108	0x108
(%eax)	0xFF
4(%eax)	0xAB
9(%eax, %edx)	0x11
260(%ecx, %edx)	0x13
0xFC(,%ecx, 4)	0xFF
(%eax, %edx, 4)	0x11

4*%ecx

addr = 4 + 0x100 = 0x104

addr = 9 + 0x100 + 0x3

0x100
+0x003
0x103
+0x009
0x10C

260 + 0x4 = 264

0x 1 0 8

256 16 1

1
0xFC
+0x04
0x100

0x100
0x00C
0x10C