11-isa-notes

ISA (p2)

Agenda

(Textbook: 4.0 through 4.1.3 again. And citing Fig 3.3 and 3.12)

- 1. The Instruction Set Architecture (Review)
- 2. Addressing Modes Revisited
- 3. Explicit Examples

---

0. Re-Orienting

The computation stack, from the human all the way down to the physical universe

The idea of an "instruction set architecture" (and the Y86 ISA in particular)

The idea of a datapath. (a given ISA may have many!)

The mapping of higher-level languages down to sequences of instructions in an ISA

The mapping of a datapath's operations down to sequential logic

---

1. The ISA (Review)

Things you can play with

How you can play with them

- Fig 4.2, 4.3, and 4.4 tell you how to assemble Y86 instruction codes.
- Fig 3.3 and 3.12 tell you some semantics from the X86 which the Y86 inherits
- Here's a **cheatsheat (https://ssl.cs.dartmouth.edu/~sws/cs51-s15/y86/cheatsheet.pdf)** I put together.

---

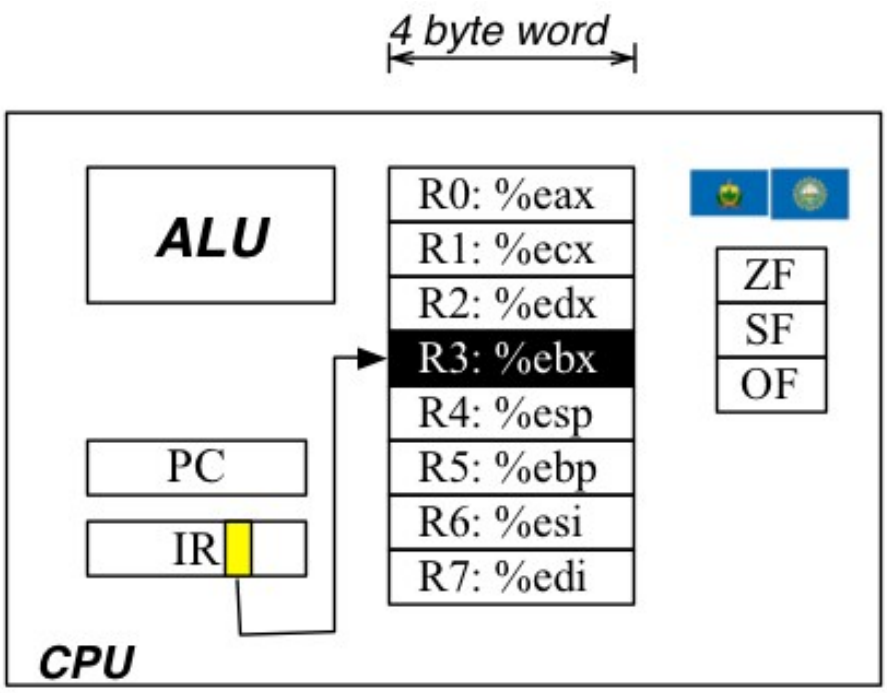
2. Addressing Modes

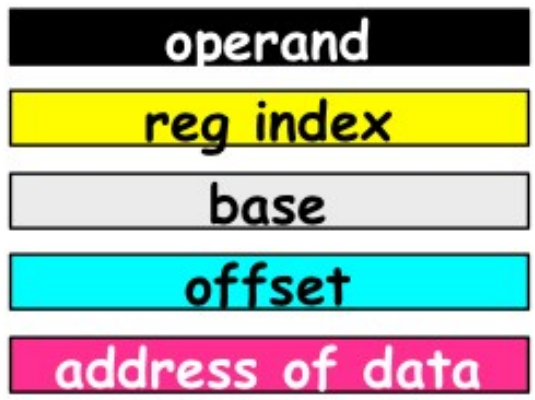
How does an opcode specify the data?

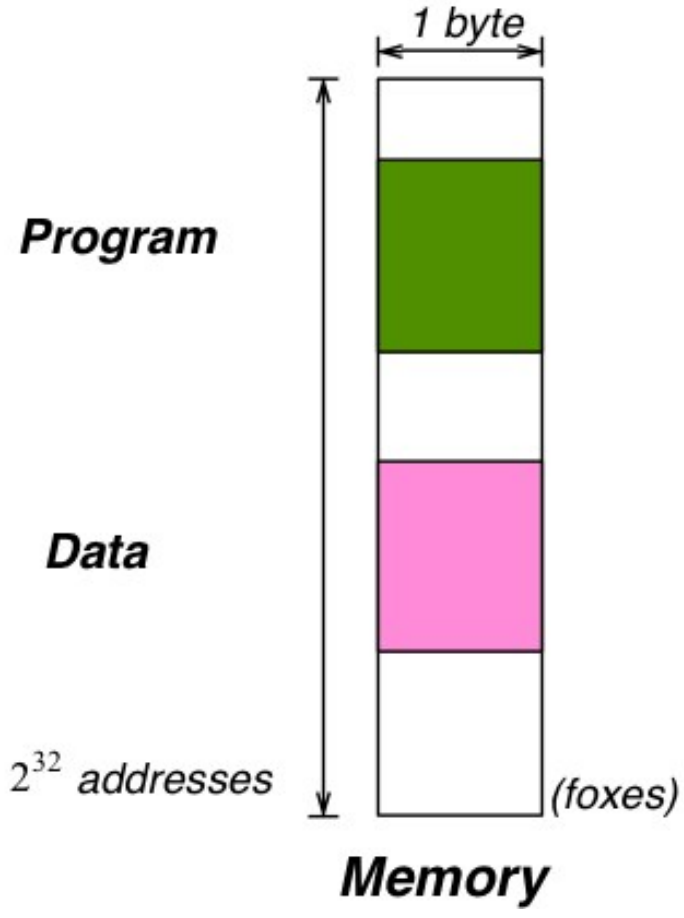
In the real world, there are many variations, and some are hard to distinguish.

Here are the principal ones in the Y86.

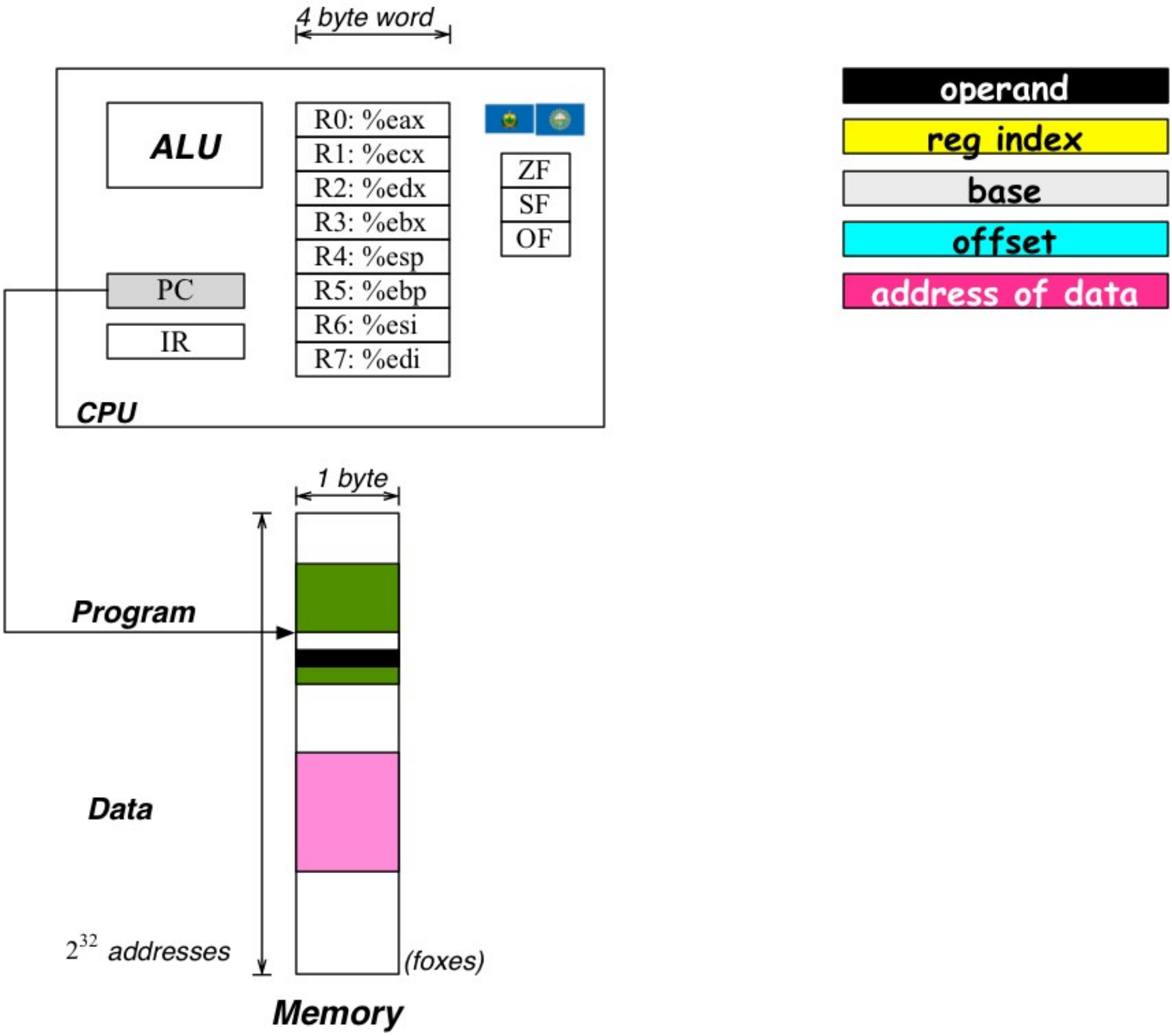
Register







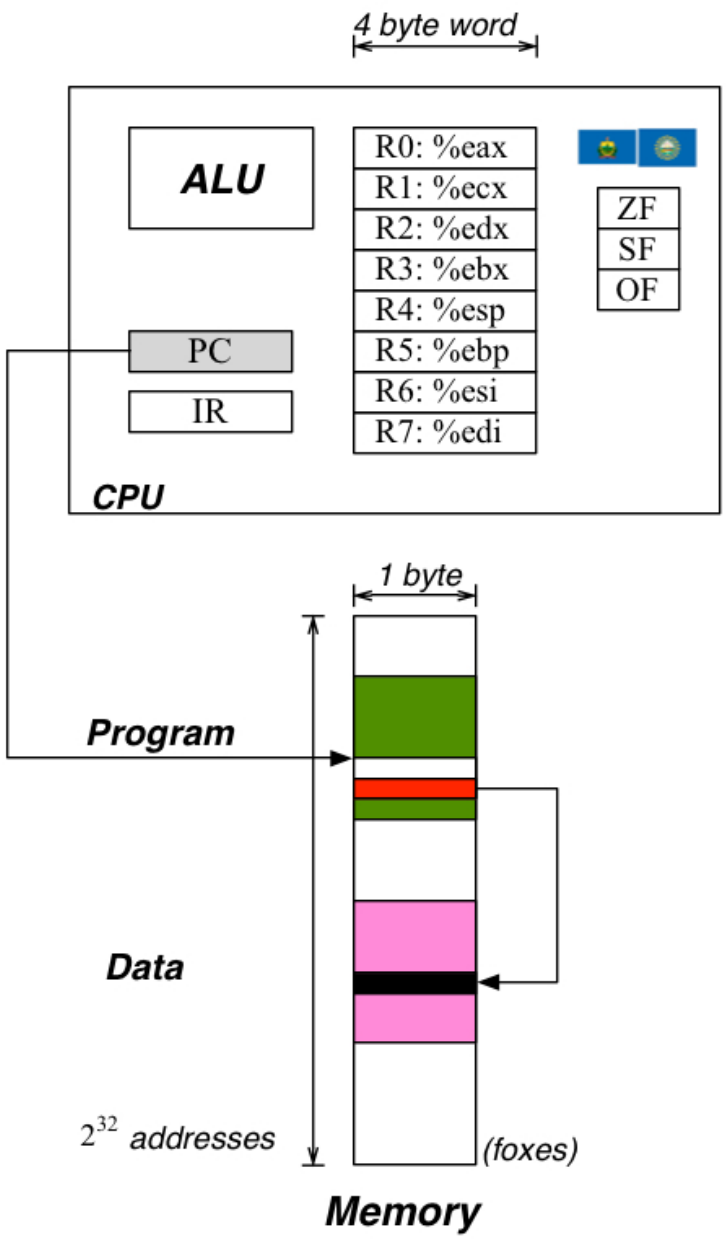
Immediate

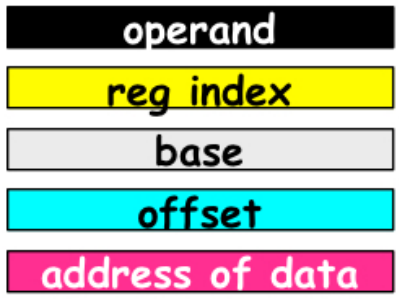


Absolute

ABSOLUTE

(a better example :)





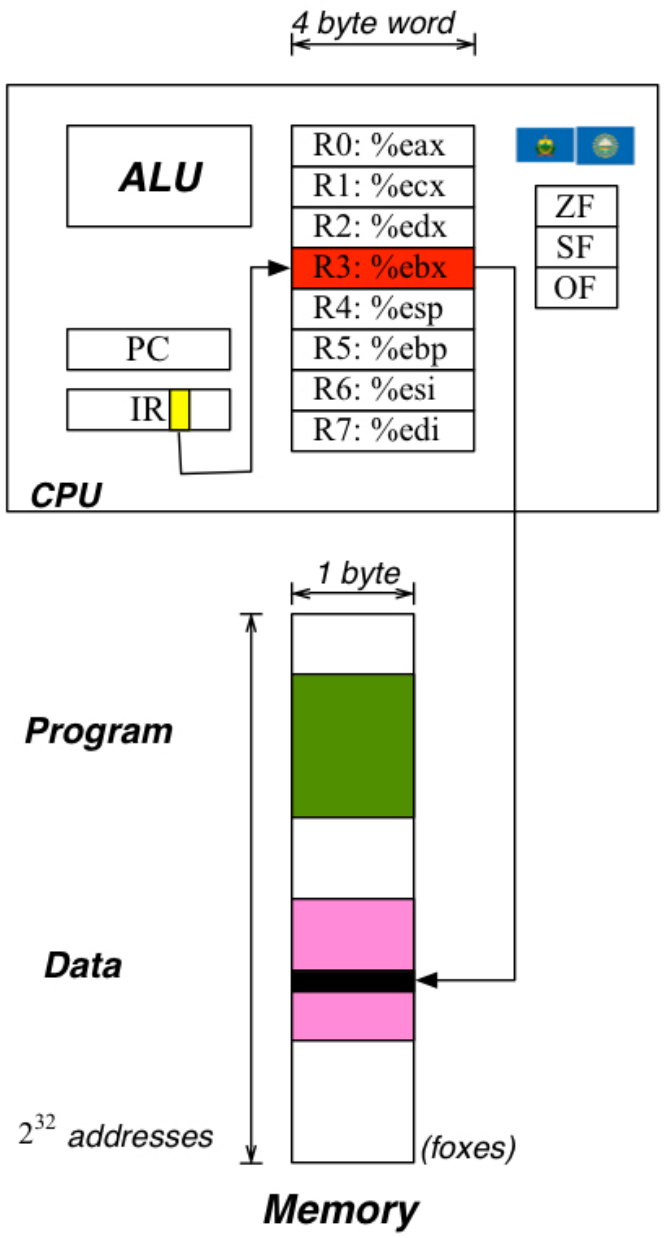
`rmmovl` **rA, D**

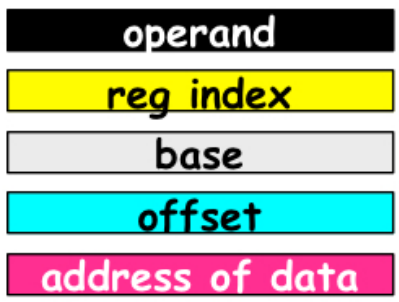
(Fig 4.2 fragment)

(move rA to the address D)

Indirect

INDIRECT (fixed :)





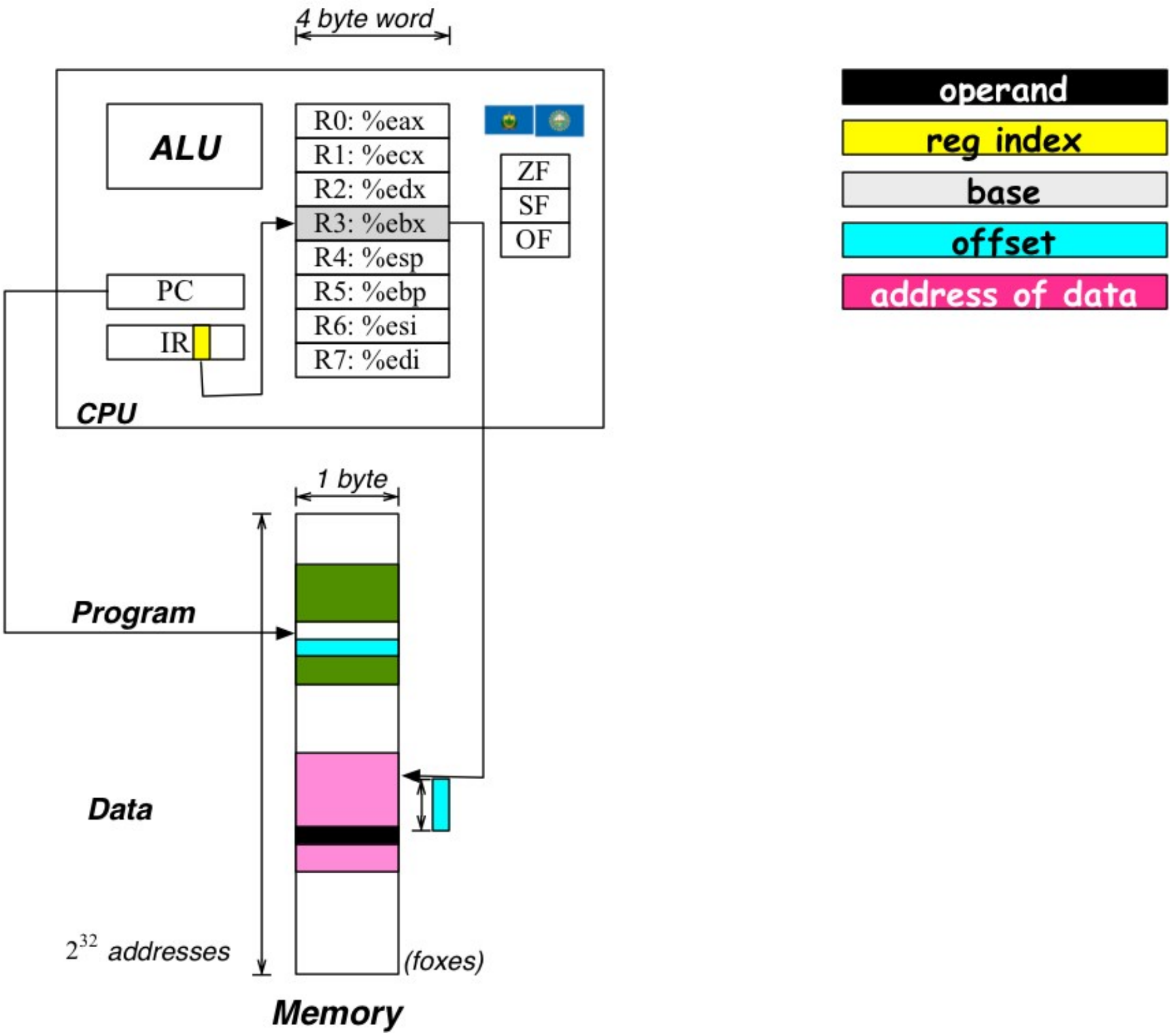
`rmmovl` **rA,** **(rB)**

4	0	rA	rB	00 00 00 00

(Fig 4.2 fragment)

Move rA to the address in rB

Base + Displacement



3. Explicit Examples

Some sample instructions:

```
        # immediate addressing!
                irmovl $0x11223344, %ecx

        # register addressing!
                rrmovl %ecx, %edx

        # indirect addressinng!
                irmovl target2, %ebx
```

```
                rmmovl %ecx, (%ebx)

        # absolute addressing!
                rmmovl %ecx, 0x50

        # base + displacement
                rmmovl %ecx, 4(%ebx)
```

Or as a program! **amodes.ys (https://ssl.cs.dartmouth.edu/~sws/cs51-s15/11-isa/amodes.zip)**

Assemble with yas , the assembler for the Y86 (available via our Y86 resources page). (The result: amodes.yo)

Note...

- how the instructions are put together
- the D = 0
- the register code of F
- the little-endian 4-byte words

Then run with ssim (also at our Y86 page)

- operations of the instructions
- what happens to the PC