Intel Assembly

Format of an assembly instruction:

```
LABEL      OPCODE   OPERANDS COMMENT
DATA1      db       00001000b   ;Define DATA1 as decimal 8
START:     mov      eax, ebx    ;Copy ebx to eax
```

LABEL:

Stores a symbolic name for the memory location that it represents.

OPCODE:

The instruction itself.

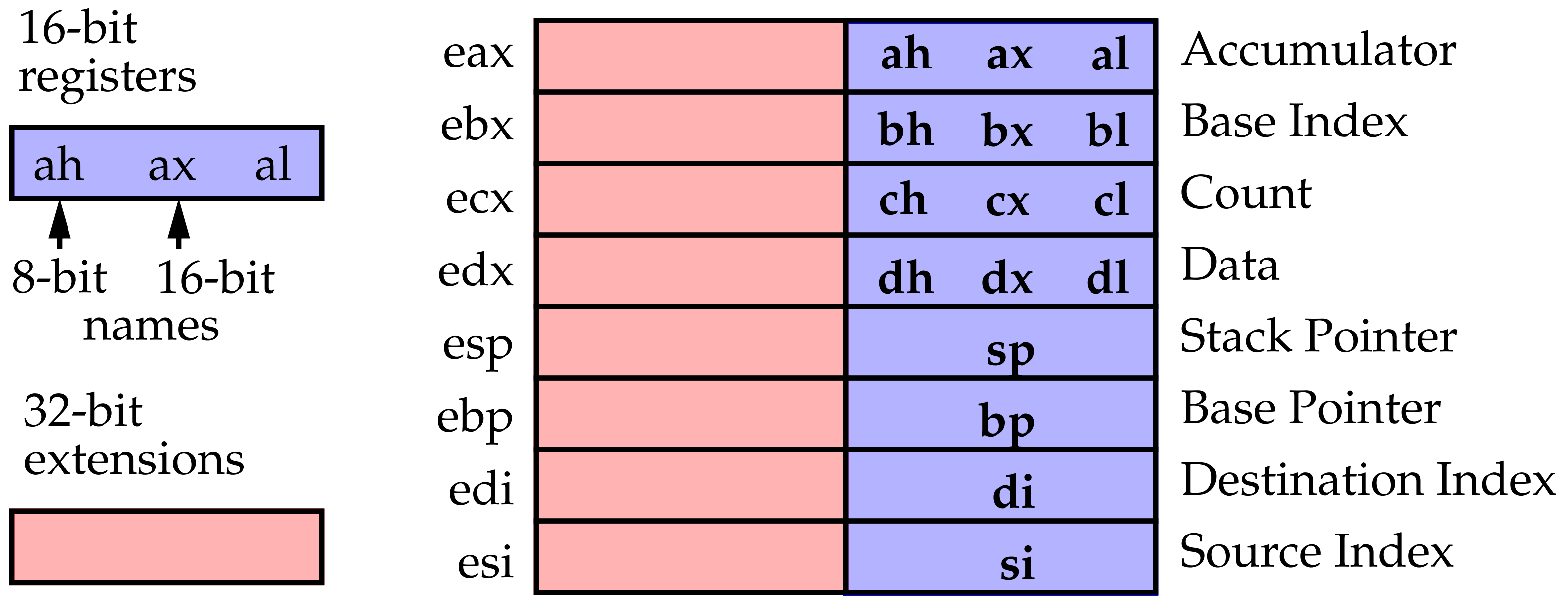
OPERANDS:

A register, an immediate or a memory address holding the values on which the operation is performed.

There can be from 0 to 3 operands.

Data Addressing Modes

Data registers:



Let’s cover the data addressing modes using the **mov** instruction.

Data movement instructions move data (bytes, words and doublewords) between registers and between registers and memory.

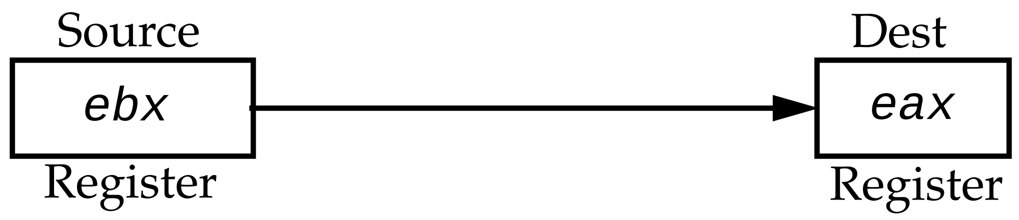
Only the **movs** (strings) instruction can have both operands in memory.

Most data transfer instructions do not change the **EFLAGS** register.

Data Addressing Modes

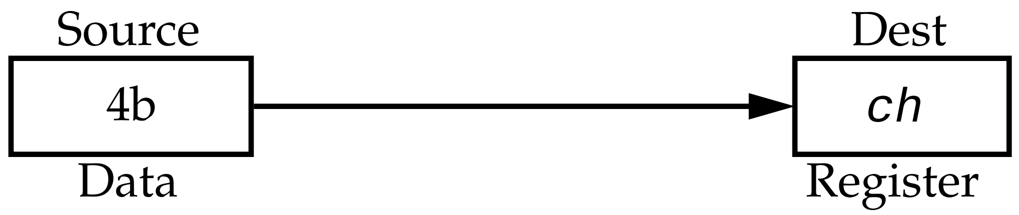
- Register

```
mov eax, ebx
```



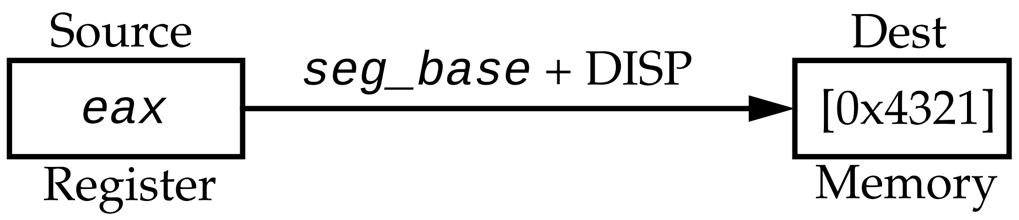
Immediate

```
mov ch, 0x4b
```



Direct (eax), Displacement (other regs)

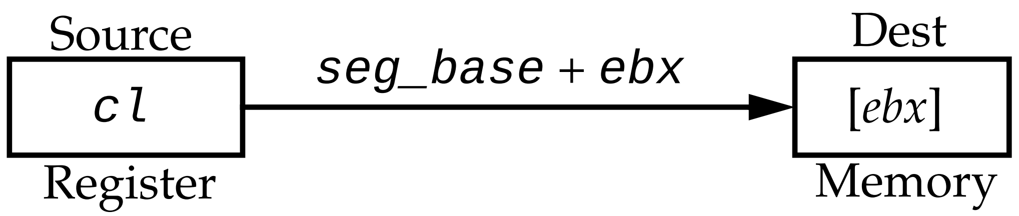
```
mov [0x4321], eax
```



Data Addressing Modes

- Register Indirect

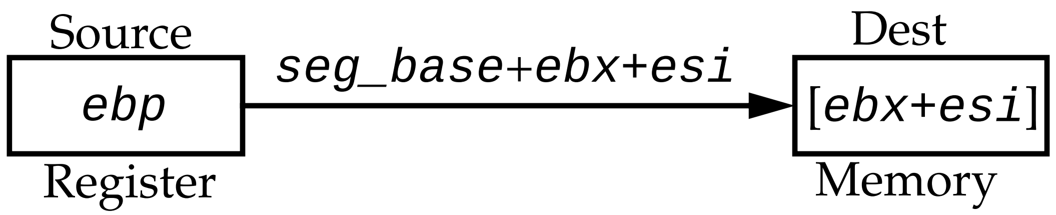
```
mov [ebx], cl
```



Any of *eax*, *ebx*, *ecx*, *edx*, *ebp*, *edi* or *esi* may be used.

- Base-plus-index

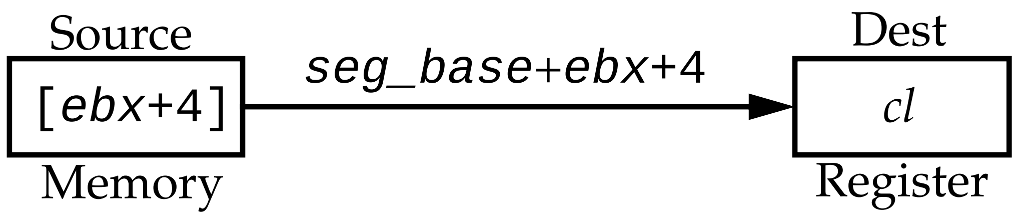
```
mov [ebx+esi], ebp
```



Any combination of *eax*, *ebx*, *ecx*, *edx*, *ebp*, *edi* or *esi*.

- Register relative

```
mov cl, [ebx+4]
```

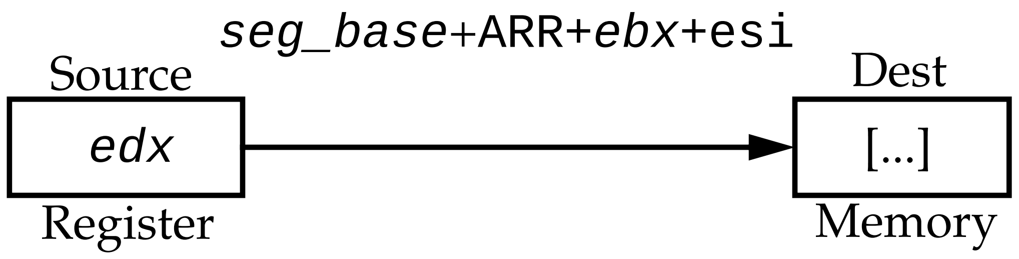


A second variation includes: **mov** *eax*, [ARR+*ebx*]

Data Addressing Modes

- Base relative-plus-index

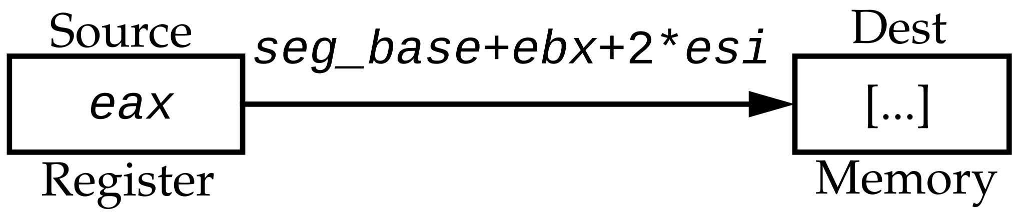
mov [ARR+*ebx*+*esi*], *edx*



A second variation includes: **mov** *eax*, [*ebx*+*edi*+4]

- Scaled-index

mov [*ebx*+2**esi*], *eax*



A second variation includes: **mov** *eax*, *ebx**2+*ecx*+offset
Scaling factors can be 2X, 4X or 8X.

Data Addressing Modes

Register addressing:

Note: **mov** really COPIES data from the source to destination register.

- Never mix an 16-bit register with a 32-bit, etc.

  For example

```
mov eax, bx      ;ERROR: NOT permitted.
```

- None of the **mov** instruction effect the EFLAGS register.

Immediate addressing:

The value of the operand is given as a constant in the instruction stream.

```
mov eax, 0x12345
```

- Use **b** for binary, **q** for octal and nothing for decimal.

- ASCII data requires a set of apostrophes:

```
mov eax, 'A'     ;Moves ASCII value 0x41 into eax.
```

Data Addressing Modes

Register and immediate addressing example:

```
          global main
          section .text   ;start of the code segment.
main:
          mov eax, 0      ;Immediate addressing.
          mov ebx, 0x0000
          mov ecx, 0
          mov esi, eax    ;Register addressing.
          ...
```

Direct addressing:

Transfers between memory and *al*, *ax* and *eax*.

Usually encoded in 3 bytes, sometime 4:

```
mov al, DATA1        ;Copies a byte from DATA1.
mov al, [0x4321]     ;Some assemblers don't allow this.
mov al, ds:[0x1234]
mov DATA2, ax        ;Copies a word to DATA2.
```

Data Addressing Modes

Displacement:

```
mov cl, DATA1        ;Copies a byte from DATA1.
mov edi, SUM         ;Copies a doubleword from SUM.
```

Displacement instructions are encoded with up to 7 bytes (32 bit register and a 32 bit displacement).

Direct and displacement addressing example:

```
                     global main
0000                 section .data
0000 10       DATA1  db     0x10
0001 00       DATA2  db     0

0000                 section .text
              main:
0017 A0 0000 R       mov al, DATA1
001A 8B 1E 0001 R    mov bx, DATA2
```

Note: Direct addressing (using **al**) requires 3 bytes to encode while Displacement (using **bx**) requires 4.

Data Addressing Modes

Register Indirect addressing:

Offset stored in a register is added to the segment register.

```
mov ecx, [ebx]
mov [edi], [ebx]
```

The memory to memory **mov** is allowed with string instructions.

Any register EXCEPT **esp** for the 80386 and up.

For **eax**, **ebx**, **ecx**, **edx**, **edi** and **esi**: The data segment is the default.

For **ebp**: The stack segment is the default.

Some versions of register indirect require special assembler directives *byte*, *word*, or *dword*

```
mov al, [edi]      ;Clearly a byte-sized move.
mov [edi], 0x10    ;Ambiguous, assembler can't size.
```

Does [*edi*] address a byte, a word or a double-word?

Use:

```
mov byte [edi], 0x10    ;A byte transfer.
```