L'Assembler 8086

Formato delle istruzioni macchina
Tempi di esecuzione

M. Rebaudengo - M. Sonza Reorda

Politecnico di Torino
Dip. di Automatica e Informatica

Formato delle istruzioni macchina

Non esistono regole generali per la traduzione delle istruzioni dal linguaggio sorgente al codice macchina.

Per ogni istruzione si hanno regole specifiche.

È tuttavia possibile fare alcune considerazioni generali.

Numero di byte

Le istruzioni macchina dell'8086 hanno una dimensione che varia da 1 a 6 byte.

Il formato prevede:

- **1 o 2 byte per specificare il codice operativo ed il modo di indirizzamento**
- **da 0 a 4 byte aggiuntivi, contenenti eventuali offset in memoria o valori immediati.**

Primo Byte

Oltre al Codice Operativo, il primo byte può contenere alcuni bit con un significato particolare:

- **W: se vale 0 l'istruzione lavora sui byte, se vale 1 lavora sulle word**
- **D: nelle istruzioni con 2 operandi uno di questi deve normalmente essere un registro; a seconda del valore di D, il registro corrisponde all'operando sorgente (D=0) o destinazione (D=1)**
- **S: compare nelle istruzioni che prevedono un operando immediato; se questo è su una word (W=1) ma il MSB è nullo, è possibile rappresentare il solo LSB ponendo S=1.**

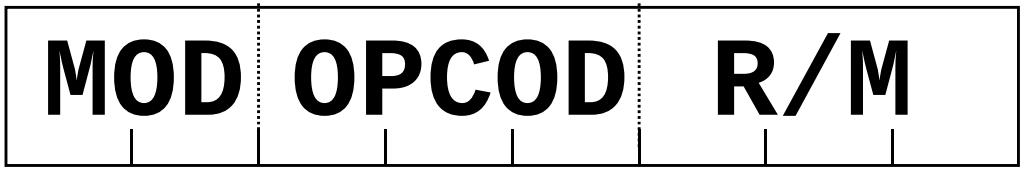
Secondo Byte

In alcune istruzioni il secondo byte è ancora destinato a specificare il codice operativo ed il modo di indirizzamento.

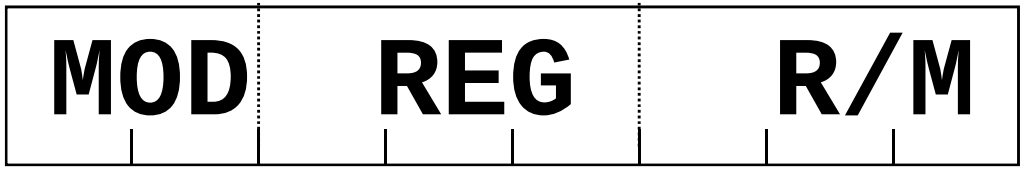
Tale byte può assumere 2 formati:

- **il primo viene utilizzato per istruzioni con un solo operando**
- **il secondo per istruzioni con due operandi; in tal caso uno dei due corrisponde al registro specificato dal campo REG:**

Formato 1



Formato 2



REG

Codice	*Registro*	
	W=1	**W=0**
000	**AX**	**AL**
001	**CX**	**CL**
010	**DX**	**DL**
011	**BX**	**BL**
100	**SP**	**AH**
101	**BP**	**CH**
110	**SI**	**DH**
111	**DI**	**BH**

MOD e R/M

Il sottocampo `MOD` specifica il *modo di indirizzamento* e se nell'istruzione è presente un *displacement*.

Il sottocampo `R/M` specifica se si usa un registro oppure una locazione in memoria.

MOD

MOD	funzione
00	usa la Tavola 1 per l'operando R/M
01	usa la Tavola 1 con displacement su 8 bit
10	usa la Tavola 1 con displacement su 16 bit
11	l'operando R/M è un registro

REG e R/M

Il sottocampo R/M (quando MOD = 11) specifica un registro:

	W=0	W=1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

R/M

Se il campo MOD contiene uno dei valori 00, 01 o 10, il sottocampo R/M assume un significato diverso, ossia specifica il tipo di indirizzamento:

MOD=00 (tavola 1)

```
000   DS:[BX+SI]
001   DS:[BX+DI]
010   SS:[BP+SI]
011   SS:[BP+DI]
100   DS:[SI]
101   DS:[DI]
110   Direct address
111   DS:[BX]
```

MOD=01 o 10 (tavola 2)

```
000   DS:[BX+SI]
001   DS:[BX+DI]
010   SS:[BP+SI]
011   SS:[BP+DI]
100   DS:[SI]
101   DS:[DI]
110   SS:[BP]
111   DS:[BX]
```

Segment Override Prefix

Per l'esecuzione di un segment override il processore utilizza un byte supplementare che precede l'istruzione ed ha il seguente formato:

0 0 1 REG 1 1 0

dove REG può assumere i seguenti valori:

0 0	**ES**
0 1	**CS**
1 0	**SS**
1 1	**DS**

Esempio: MOV

Table 2. Instruction Set Summary

Mnemonic and Description	Instruction Code			
DATA TRANSFER				
MOV = Move:	**7 6 5 4 3 2 1 0**	**7 6 5 4 3 2 1 0**	**7 6 5 4 3 2 1 0**	**7 6 5 4 3 2 1 0**
Register/Memory to/from Register	1 0 0 0 1 0 d w	mod reg r/m		
Immediate to Register/Memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1
Immediate to Register	1 0 1 1 w reg	data	data if w = 1	
Memory to Accumulator	1 0 1 0 0 0 0 w	addr-low	addr-high	
Accumulator to Memory	1 0 1 0 0 0 1 w	addr-low	addr-high	
Register/Memory to Segment Register	1 0 0 0 1 1 1 0	mod 0 reg r/m		
Segment Register to Register/Memory	1 0 0 0 1 1 0 0	mod 0 reg r/m		

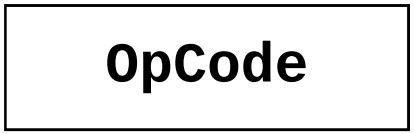
Esempi

Verranno considerati 7 casi esemplificativi di istruzioni, aventi formato tra loro diverso.

Istruzioni su 1 byte (I)

Questo formato è tipico delle istruzioni senza operandi.

Formato:



Esempio:

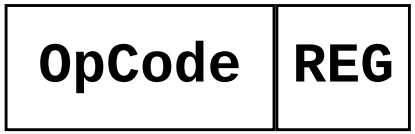
L'istruzione NOP è codificata come

1001 0000

Istruzioni su 1 byte (II)

Questo formato è tipico delle istruzioni con un solo operando, corrispondente ad un registro.

Formato:



Esempio:

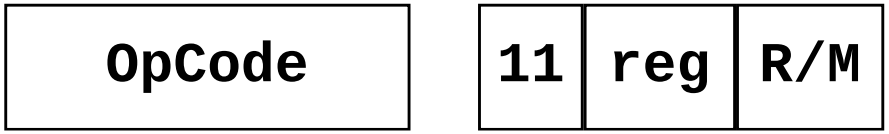
L'istruzione PUSH, qualora lavori su un registro, è codificata come

01 010 REG

Istruzioni su 2 byte (I)

Questo formato è tipico delle istruzioni con due operandi, corrispondenti entrambi ad un registro.

Formato:



Esempio:

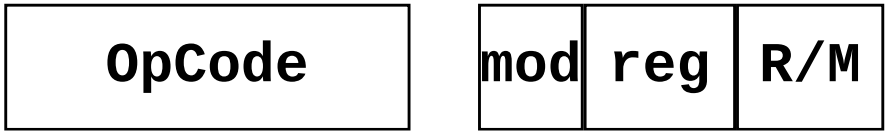
L'istruzione MOV AX, BX è codificata come

10 00 10 1 1 11 000 011

Istruzioni su 2 byte (II)

Questo formato è tipico delle istruzioni con due operandi, di cui è un registro, e l'altro risiede in memoria, e ad esso si accede tramite indirizzamento indiretto con registro.

Formato:



Esempio:

L'istruzione MOV AX, [BX] è codificata come

10 00 10 1 1 00 000 111

Istruzioni su 3 byte

Questo formato è tipico delle istruzioni con due operandi, di cui è un registro, e l'altro è un valore immediato.

Formato:

Esempio:

L'istruzione MOV AX, imm. è codificata su 3 byte.

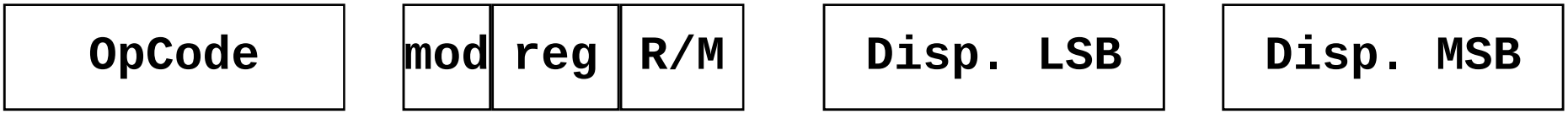
Nota

Qualora i bit S e W valgano 11, il dato può essere rappresentato su un solo byte ed esteso poi a 2 byte in fase di esecuzione.

Istruzioni su 4 byte (I)

Questo formato è tipico delle istruzioni con due operandi, di cui il primo è un registro, e l'altro risiede in memoria, e ad esso si accede specificando un offset.

Formato:



Esempio:

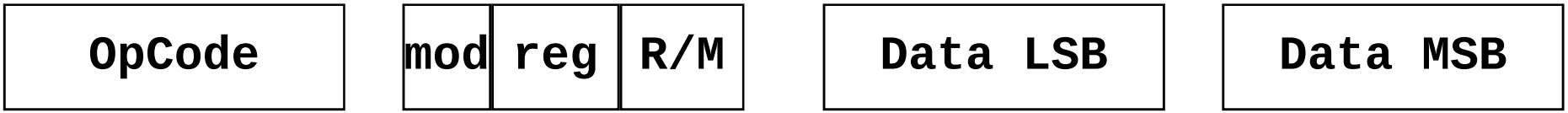
L'istruzione `MOV AX, var` è codificata come

10 00 10 11 00 000 110 offset LSB offset MSB

Istruzioni su 4 byte (II)

Questo formato è tipico delle istruzioni con due operandi, di cui è un registro, e l'altro è un operando immediato.

Formato:



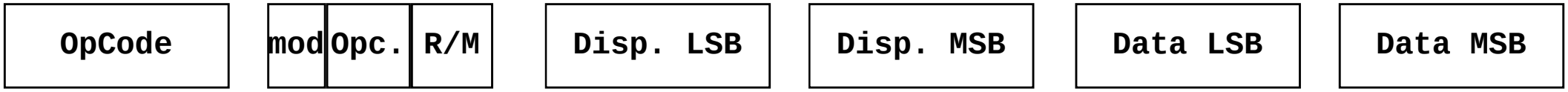
Esempio:

L'istruzione `MOV [BX], imm.` è codificata su 4 byte.

Istruzioni su 6 byte

Questo formato è tipico delle istruzioni con due operandi, di cui uno risiede in memoria, e l'altro è un operando immediato.

Formato:



Esempio:

L'istruzione `MOV var, imm.` è codificata su 6 byte.

Esempio

opcode D WMOD REG R/M

100010 1 1 11 101 100

Opcode = MOV

D = trasferimento a REG

W = word

MOD = R/M è un registro

REG = BP

R/M = SP

MOV BP, SP

Esempio

opcode W MOD REG R/M LOW DISP HIGH DISP

1100011 1 10 000 111 00000000 00001000

LOW DATA HIGH DATA

00110100 00010010

Opcode = MOV immediata

W = Word

MOD = 16 bit displacement

REG = 000 (non usata nell'indirizzamento immediato)

R/M = DS:[BX]

Displacement = 1000H

Data = 1234H

MOV WORD PTR [BX+1000H], 1234H

Tempi di esecuzione

Il tempo di esecuzione di un'istruzione dipende

- **dalla frequenza di clock**
- **dal tipo dell'istruzione**
- **dalla posizione degli operandi (in un registro, immediato, in memoria)**
- **dall'allineamento degli operandi in memoria.**

Il tempo richiesto può essere così scomposto:

- **tempo per il calcolo dell'EA dell'eventuale operando in memoria**
- **tempo per l'accesso a tale operando**
- **tempo per l'esecuzione.**

Numero di colpi di clock

I manuali forniscono per ciascuna istruzione il numero di colpi di clock necessari.

Per ottenere il tempo, tale numero va moltiplicato per il periodo del clock.

Tempo per l'esecuzione

Dipende dall'istruzione, dagli operandi e in alcuni casi dal loro valore.

Esempio

Execution times for typical instructions (in clock cycles)

instruction	register-register	register immediate	register-memory	memory-register	memory-immediate
mov	2	4	8+EA	9+EA	10+EA
ALU	3	4	9+EA,	16+EA,	17+EA
jump	*register* => 11 ; *label* => 15 ; *condition,label* => 16				
integer multiply	70~160 (depending on operand *data* as well as size) plus EA				
signed integer divide	80~190 (depending on operand *data* as well as size) plus EA				

Calcolo dell'EA

Indirizzamento	*# clock*
Diretto	**6**
Register Indirect	**5**
Register Relative	**9**
Based Indexed	
[BP]+[DI]	**7**
[BX]+[SI]	**7**
[BP]+[SI]	**8**
[BX]+[DI]	**8**
Based Indexed Relative	
[BP]+[DI]+disp	**11**
[BX]+[SI]+disp	**11**
[BP]+[SI]+disp	**12**
[BX]+[DI]+disp	**12**

Tempo per l'accesso all'operando

Se l'operando in memoria è una word posta ad un indirizzo dispari, l'8086 richiede 4 colpi di clock in più per ogni accesso in memoria.

Se quindi l'operando in memoria coincide con il risultato, si dovranno aggiungere 8 colpi di clock.

Esempio 1

```
ADD AX, BX
```

# colpi di clock richiesti:	**3**
frequenza di clock	**5 MHz**
tempo richiesto	**600 nsec**

Esempio 2

`ADD AX, [BX+SI]+4`

# colpi di clock richiesti per l'esecuzione	**9**
# colpi di clock richiesti per il calcolo dell'EA	**11**
# colpi di clock aggiuntivi se l'operando è ad un indirizzo dispari	**4**
frequenza di clock	**5 MHz**
tempo richiesto	**(9+11+4)*200nsec=4,8 μsec**

Esempio 3

`ADD [BX+SI]+4, AX`

 # colpi di clock richiesti per l'esecuzione 9

 # colpi di clock richiesti per il calcolo dell'EA 11

 # colpi di clock aggiuntivi se l'operando è

ad un indirizzo dispari 4

frequenza di clock 5 MHz

tempo richiesto (9+11+4+4)*200nsec=5,6 μsec