Узагальнене визначення кодів команд

Загальний формат команди процесора Intel

Команда може мати до шести полів:
1. Префікси – від нуля до чотирьох однобайтових префіксів.
2. Код – один або два байти, що визначають команду.
3. ModRM – 1 байт способу адресування (якщо він потрібний), який описує операнди:
   (нумерація бітів справа наліво починаючи з нуля)
   біти 7-6: поле MOD – режим адресування;
   біти 5-3: поле R/O – або визначає регістр, або є продовженням коду команди;
   біти 2-0: поле R/M – або визначає регістр, або разом з MOD – режим адресування.
4. 6):
   біти 7-6: S – коефіцієнт масштабування;
   біти 5-3: I – індексний регістр;
   біти 2-0: B – регістр бази.
5. Зміщення – 0, 1, 2 або 4 байти.
6. Безпосередній операнд – 0, 1, 2 або 4 байти.

Значення полів коду команди

У кодах деяких команд зустрічаються спеціальні біти і групи бітів, котрі позначають w, s, d, reg, sreg, cond:
w = 0, якщо команда працює з байтами;
w = 1, якщо команда працює з словами або подвійними словами;
s = 0, якщо безпосередній операнд записаний в команді повністю;
s = 1, якщо безпосередній операнд є молодшим байтом більшого операнда і має розглядатись як число зі знаком;
d = 0, якщо код джерела записаний в полі R/O, а приймача – в полі R/M;
d = 1, якщо код джерела записаний в полі R/M, а приймача – в полі R/O.

Поле reg визначає потрібний регістр і має довжину 3 біти:
000 – AL / AX / EAX / ST(0) / MM0 / XMM0
001 – CL / CX / ECX / ST(1) / MM1 / XMM1
010 – DL / DX / EDX / ST(2) / MM2 / XMM2
011 – BL / BX / EBX / ST(3) / MM3 / XMM3
100 – AH / SP / ESP / ST(4) / MM4 / XMM4
101 – CH / BP / EBP / ST(5) / MM5 / XMM5
110 – DH / SI / ESI / ST(6) / MM6 / XMM6
111 – BH / DI / EDI / ST(7) / MM7 / XMM7

Поле sreg визначає потрібний сегментний регістр:
000 – ES
001 – CS
010 – SS
011 – DS
100 – FS
101 – GS

Поле cond визначає умову для команд Jcc, CMOVcc, SETcc, FCMOVcc. Воно має такі значення для різних команд:

0000 – O
0001 – NO
0010 – C / B / NAE
0011 – NC / NB / AE
0100 – E / Z
0101 – NE / NZ
0110 – BE / NA
0111 – NBE / A
1000 – S
1001 – NS
1010 – P / PE
1011 – NP / PO
1100 – L / NGE
1101 – NL / GE
1110 – LE / NG
1111 – NLE / G

Значення полів байта ModRM способу адресування

Поле R/O (біти 5-3) має або додаткові три біти коду команди, або код операнда, який є регістром. В таблицях команд другий випадок позначають reg, а в першому записують потрібні біти.
Поля MOD (біти 7-6) і R/M (біти 2-0) визначають операнд, який може бути як регістром, так і коміркою пам'яті:
MOD = 11, якщо використовується регістрове адресування і поле R/M має код регістра reg;
MOD = 00, якщо використовується адресування в пам'яті без зміщення ([BX+SI] або [EDX]), тобто неявне адресування через регістри;
MOD = 01, якщо використовується адресування в пам'яті з 8-бітовим зміщенням (var[BX+SI] чи [BX+SI]+число);
MOD = 10, – те ж саме, що й MOD=01, тільки зміщення 16-бітове або 32-бітове.

Значення поля R/M *відрізняється* в 16- і 32-бітових режимах.
R/M в 16-бітовому режимі:
000 – [BX + SI]
001 – [BX + DI]
010 – [BP + SI]
011 – [BP + DI]
100 – [SI]
101 – [DI]
110 – [BP] (крім MOD=00 – в цьому випадку після байта ModRM записується 16-бітове зміщення, тобто використовується пряме адресування: ADD DX,Table)
111 – [BX]
R/M в 32-бітовому режимі:
000 – [EAX]
001 – [ECX]
010 – [EDX]
011 – [EBX]
100 – використовується байт SIB
101 – [EBP] (крім MOD=00 – в цьому випадку після байта ModRM записується 32-бітове зміщення, тобто використовується пряме адресування: ADD EDX,Table)
110 – [ESI]
111 – [EDI]

Значення полів байта SIB (розширення ModRM)

Якщо такий байт присутній в команді, тоді повна адреса операнда в пам'яті обчислюється за виразом [B]+[I]*S+зміщення.
Значення поля S:
00 – не використовується (тобто, множення на 1)
01 – множення на 2
10 – множення на 4
11 – множення на 8

Значення полів I та B:
(I – регістр, який вважається індексним і множиться на S; B – регістр бази, який не множиться)
000 – EAX
001 – ECX
010 – EDX
011 – EBX
100 – для I – індекса немає; для B – ESP
101 – для I – EBP; для B – EBP, лише при MOD=01 або 10, при MOD=00 бази немає і тоді після байта SIB записується 32-бітове зміщення
110 – ESI
111 – EDI

Префікси

Всі префікси виконуються за 1 такт і мають розмір 1 байт:
0F0h: LOCK
0F2h: REPNE / REPNZ
0F3h: REP / REPE / REPZ
2Eh: CS:
36h: SS:
3Eh: DS:
26h: ES:
64h: FS:
65h: GS:
66h: OS
67h: AS

Повна форма адресування операндів в пам'яті

У повній формі адресування операндів відбувається по базі з індексуванням і масштабуванням. Інші способи адресування операндів в пам'яті можна розглядати як часткові випадки повної форми адресування. Адреса операнда обчислюється як сума трьох доданків: значення базового регістра; значення індексного регістра, помноженого на коефіцієнт (масштаб) 1, 2, 4 або 8; зміщення. Наприклад:

```
MOV  EAX,[ECX][ESI*4]+10
ADD  Temp[EBX+EDI*2],EAX
```

Зміщення може бути байтом або подвійним словом. Якщо ESP або EBP заданий як базовий регістр, то селектор сегмента операнда визначається за замовчуванням за регістром SS, у всіх інших випадках – за регістром DS.

Схему обчислення адреси операнда за повною формою можна подати таким рисунком:

	EAX		EAX			
CS:	EBX		EBX			
SS:	ECX		ECX		1	
DS:	EDX	+	EDX	*	2	+ зміщення
ES:	EBP		EBP		4	
FS:	ESP		ESI		8	
GS:	EDI		EDI			
	ESI					

Приклади трансляції команд в 32-бітовому режимі

Нижче подано декілька прикладів трансляції окремо взятих команд. Ліворуч від команди записаний шістнадцятковий код машинної команди – так, як друкує асемблер, без врахування перестановки байтів зміщення і безпосередніх операндів. В наступному рядку після команди подано зображення байтів коду операції, ModRM і SIB в двійковому коді з розділенням полів (для ModRM і SIB), а байти зміщення і безпосередніх операндів записані в справжньому порядку.

Уважно порівняйте подібні зовні команди після трансляції в машинні коди, щоб зрозуміти різницю і особливості трансляції.

Вважаємо, що адреса комірки Temp дорівнює 270 десяткове (шістнадцяткова 10E).

```
2B F1           SUB  ESI,ECX
; 00101011  11 110 001
2B CE           SUB  ECX,ESI
; 00101011  11 001 110
2B 31           SUB  ESI,[ECX]
; 00101011  00 110 001
83 C1 23        ADD  ECX,35
; 10000011  11 000 001  00100011
81 C1 0000041A  ADD  ECX,1050
; 10000001  11 000 001  00011010 00000100 00000000 00000000
F7 EA           IMUL     EDX
; 11110111  11 101 010
0F AF CA        IMUL     ECX,EDX
; 00001111  10101111  11 001 010
A1 0000010E R       MOV  EAX,Temp
; 10100001  00001110 00000001 00000000 00000000
8B 15 0000010E R    MOV  EDX,Temp
; 10001011  00 010 101  00001110 00000001 00000000 00000000
03 96 0000010E R    ADD  EDX,Temp[ESI]
; 00000011  10 010 110  00001110 00000001 00000000 00000000
03 56 05        ADD  EDX,[ESI+5]
; 00000011  01 010 110  00000101
03 54 33 05     ADD  EDX,[ESI][EBX]+5
; 00000011  01 010 100  00 110 011  00000101
03 54 73 05     ADD  EDX,[ESI*2][EBX]+5
; 00000011  01 010 100  01 110 011  00000101
```

```
03 54 5E 05     ADD EDX,[ESI][EBX*2]+5
; 00000011  01 010 100  01 011 110  00000101
01 93 0000010E R         ADD  Temp[EBX],EDX
; 00000001  10 010 011  00001110 00000001 00000000 00000000
01 14 5D 0000010E R      ADD  Temp[EBX*2],EDX
; 00000001  00 010 100  01 011 101  00001110 00000001 00000000
               00000000
01 94 5E 0000010E R      ADD  Temp[ESI][EBX*2],EDX
; 00000001  10 010 100  01 011 110  00001110 00000001 00000000
               00000000
83 84 5E 0000010E R 23   ADD  Temp[ESI][EBX*2],35
; 10000011  10 000 100  01 011 110  00001110 00000001 00000000
               00000000  00100011
81 84 5E 0000010E R 0000041A  ADD  Temp[ESI][EBX*2],1050
; 10000001  10 000 100  01 011 110  00001110 00000001 00000000
               00000000  00011010 00000100 00000000 00000000
```