Arhitectura Sistemelor de Calcul

Lect. Dr. Șotropa Diana
diana.sotropa@ubbcluj.ro

Facultatea de Matematică și Informatică
Universitatea Babeș-Bolyai

Reprezentarea instrucțiunilor mașină

Reprezentarea instrucțiunilor mașină

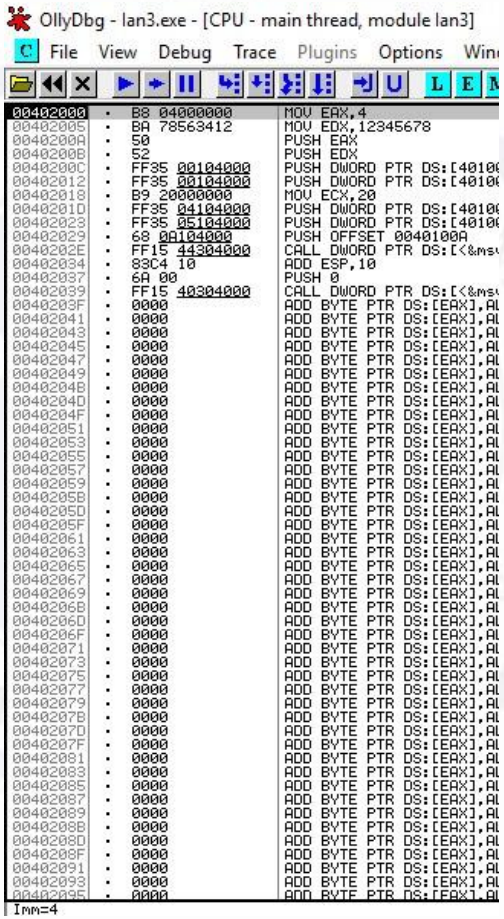
- O instrucţiune maşină x86 reprezintă o secvenţă de 1 până la 15 octeţi, care prin valorile lor specifică o operaţie de executat, operanzii asupra cărora va fi aplicată, precum şi modificatori suplimentari care controlează modul în care aceasta va fi executată.
- O instrucţiune maşină x86 are maximum doi operanzi.
- Pentru cele mai multe dintre instrucţiuni, cei doi operanzi poartă numele de **sursă**, respectiv **destinaţie**.
- Dintre cei doi operanzi, maximum unul se poate afla în memoria RAM. Celălalt se află fie într-un registru al EU, fie este o constantă întreagă.
- Astfel, o instrucţiune are forma:

```
numeinstrucţiune destinaţie, sursă
```

- Formatul intern al unei instrucţiuni este variabil, el putând ocupa între 1 şi 15 octeţi, având următoarea formă generală de reprezentare

`[prefixe]+cod+[ModR/M]+[SIB]+[deplasament]+[imediat]`

Reprezentarea instrucțiunilor mașină



Reprezentarea instrucțiunilor mașină

Auxiliare

- 18. Reprezentarea instructiunilor masina - 01_Intel x86 Assembler Instruction Set Opcode Table.pdf
- 18. Reprezentarea instructiunilor masina - 02_Opcode_Extensions_Table.pdf
- 18. Reprezentarea instructiunilor masina - 03_Addressing Method Codes.pdf
  - lămureşte cum să înţelegem OPERANZII instrucţiunilor din tabelele SetOpcode Table
- 18. Reprezentarea instructiunilor masina - 04_05_Intel_MODRM_SIB_Tables.pdf
  – lămuresc modul de CODIFICARE în cadrul unei instrucţiuni (vezi coloana 2 din OllyDbg) al octeţilor Mod R/M şi SIB

Reprezentarea instrucțiunilor mașină

Formatul intern al unei instrucţiuni

[prefixe]+cod+[Mod R/M]+[SIB]+[deplasament]+[imediat]

Prefix	Cod	Mod R/M	SIB	Deplasament	Imediat
0, 1, 2, 3 sau 4	1, 2 sau 3	0 sau 1	0 sau 1	0, 1, 2 sau 4	0, 1, 2 sau 4
Număr de octeți					

Prefixe pentru instrucțiuni de manipulare a șirurilor	Prefix de supra-scriere a adresei	Prefix de supra-scriere a operanzilor	Prefixul de supra-scriere a segmentului
0 sau 1	0 sau 1	0 sau 1	0 sau 1
Număr de octeți			

7	6	5	4	3	2	1	0
MOD		REG / OPCODE			R/M		

7	6	5	4	3	2	1	0
SCALĂ		INDEX			BAZĂ		

Reprezentarea instrucțiunilor mașină

- Prefixele controlează modul în care o instrucţiune se execută. Acestea sunt opţionale (0 până la maximum 4) şi ocupă câte un octet fiecare.
- De exemplu, acestea pot solicita execuţia repetată (în buclă) a instrucţiunii curente sau pot bloca magistrala de adrese pe parcursul execuţiei pentru a nu permite accesul concurent la operanzi şi rezultate.
- Operaţia care se va efectua este identificată prin intermediul a 1 sau 2 octeţi de cod (opcode), aceştia fiind **singurii octeţi obligatoriu prezenţi**, indiferent de instrucţiune.

Prefixe de instrucțiuni

- **Prefixele de instrucţiuni** sunt **construcţii ale limbajului de asamblare** care:
  - apar opţional în componenţa unei linii sursă (*prefixe explicite*) sau a formatului intern al unei instrucţiuni (*prefixe generate în mod implicit de către asamblor în două situaţii*)
  - modifică comportamentul standard al acelor instrucţiuni (*în cazul prefixelor explicite*)
  - semnalează procesorului modificarea dimensiunii implicite de reprezentare a operanzilor sau/şi a adreselor, dimensiuni stabilite prin directive de asamblare (*BITS 16 si BITS 32*).

Prefixe de instrucțiuni

- Instrucțiunile arhitecturii x86 pot avea maxim 4 prefixe
- Fiecare prefix ajustează interpretarea codului de operație (opcode):

1. Prefixe pentru instrucțiuni de manipulare a șirurilor (prefixe specificate **explicit** de către programator!)
2. Prefixul de suprascriere a segmentului (Segment override prefix)
3. Prefix de suprascriere a operanzilor
4. Prefix de suprascriere a adresei

Prefixe de instrucțiuni

Prefixe pentru instrucțiuni de manipulare a șirurilor

- Prefixe pentru instrucțiuni de manipulare a șirurilor (prefixe specificate **explicit** de către programator!):

```
F3h = REP, REPE
F2h = REPNE
```

unde:

- **REP** repetă instrucțiunea de numărul de ori specificat prin contorul de iterații **ECX**.
- **REPE** și **REPNE** permit terminarea buclei în funcție de valoarea **ZF** (Zero Flag) din CPU.

Prefixe de instrucțiuni

Prefixe pentru instrucțiuni de manipulare a șirurilor

Prefix (hex)	Denumire folosită	Instrucțiuni aplicabile	Observații
0xF3	REP	MOVS, LODS, STOS, INS, OUTS	Aceste instrucțiuni nu afectează flag-urile.
0xF3	REPE / REPZ	CMPS, SCAS	Repetă instrucțiunea atâta timp cât ZF = 1.
0xF2	REPNE / REPNZ	CMPS, SCAS	Repetă instrucțiunea atâta timp cât ZF = 0. Nu este documentat pentru alte instrucțiuni.

Prefixe de instrucțiuni

Prefixe pentru instrucțiuni de manipulare a șirurilor

- **`REP MOVSB F3:A4`**

  Instrucțiuni asociate pentru manipularea șirurilor sunt:
  - **`MOVS`** – mută șir
  - **`STOS`** – stochează șir
  - **`SCAS`** – scanează șir
  - **`CMPS`** – compară șir, etc.
- Vezi și programul exemplu pentru manipularea șirurilor: **http://www.c-jump.com/CIS77/samples/rep_movsb.htm**

Prefixe de instrucțiuni

Prefix de suprascriere a segmentului

- **Prefixul de suprascriere a segmentului (Segment override prefix)** face ca accesul la memorie să folosească segmentul specificat, în locul segmentului implicit desemnat pentru operandul instrucțiunii.
- Acestea sunt prefixe specificate **explicit** de către programator
  - `2Eh = CS`
  - `36h = SS`
  - `3Eh = DS`
  - `26h = ES`
  - `64h = FS`
  - `65h = GS`
- Exemplu:
```
26 : D7
ES xlat ; accesul la memorie se face prin (ES:EBX).
```

Prefixe de instrucțiuni

Prefixul de suprascriere a segmentului

Instrucțiune	Cod mașină	Explicații
mov eax, [ebx]	8B 03	Acces implicit prin segmentul DS
mov eax, [SS:ebx]	36:8B 03	Prefix explicit SS (36)
mov ax, [DS:ebx]	66 3E:66:8B 03	Prefix explicit DS (3E) Prefix de mărime operand 66 Cod MOV Gv, Ev - 8B Cod EBX - 03
mov eax, [CS:ebx]	2E:8B 03	Prefix explicit CS (2E)
ES LODSB	26:AC	LODS byte ptr ES:[ESI]
ES CMPSB	26:A6	CMPS byte ptr ES:[ESI], byte ptr ES:[EDI]
ES STOSB	26:AA	STOS byte ptr ES:[EDI] – prefix de segment inutil (superfluous)
MOVSB	A4	MOVS byte ptr ES:[EDI], byte ptr DS:[ESI]
ES MOVSB	26:A4	MOVS byte ptr ES:[EDI], byte ptr ES:[ESI]
ES SCASB	26:AE	SCAS byte ptr ES:[EDI] – prefix de segment inutil (superfluous)

Prefixe de instrucțiuni

Prefix de suprascriere a operanzilor

- **Prefixul de suprascriere a operanzilor – 66h** - modifică dimensiunea datelor așteptată de modul implicit al instrucțiunii, de exemplu din **16 biți în 32 biți** și invers
- apare ca rezultat al unor moduri particulare de utilizare a instrucţiunilor, utilizări care vor provoca apariţia acestor prefixe la nivelul formatului intern al instrucţiunii.
- Aceste prefixe NU se precizează EXPLICIT prin cuvinte cheie sau rezervate ale limbajului de asamblare.

Prefixe de instrucțiuni

Prefix de suprascriere a operanzilor

Mod implicit	Instrucțiune	Cod mașină	Explicații
32 biți	cbw	66:98	Rezultatul este pe 16 biți (AX), deci se folosește prefixul 66h
32 biți	cwd	66:99	Rezultatul este format din 2 registre pe 16 biți (DX:AX), deci se folosește 66h
32 biți	cwde	98	Se respectă modul default pe 32 biți, rezultat în EAX
32 biți	push ax	66:50	Se încarcă pe stivă o valoare pe 16 biți, stiva fiind organizată implicit pe 32 biți
32 biți	push eax	50	Consistent cu modul default (32 biți)
32 biți	mov ax, a	66:B8 0010	Rezultatul este pe 16 biți, deci se folosește prefixul 66h
16 biți	cbw	98	Rezultatul este pe 16 biți (AX), nu este necesar prefixul
16 biți	cwd	99	Rezultatul este format din 2 registre pe 16 biți (DX:AX), nu este necesar prefixul
16 biți	cwde	66:98	Nu se respectă modul default pe 16 biți, rezultatul se pune în EAX, deci se folosește 66h

Prefixe de instrucțiuni

Prefix de suprascriere a adresei

- **Prefixul de suprascriere a adresei – 67h** - modifică dimensiunea adresei așteptată de modul implicit al instrucțiunii. O adresă pe **32 de biți** poate fi comutată la **16 biți** și invers.
- apare ca rezultat al unor moduri particulare de utilizare a instrucţiunilor, utilizări care vor provoca apariţia acestor prefixe la nivelul formatului intern al instrucţiunii.
- Aceste prefixe NU se precizează EXPLICIT prin cuvinte cheie sau rezervate ale limbajului de asamblare.

Prefixe de instrucțiuni

Prefix de suprascriere a adresei

Mod implicit	Instrucțiune	Cod mașină	Explicații
32 biți	mov eax, [bx]	67:8B07	Adresare pe 16 biți prin DS:[BX], deci se folosește prefixul 67h
16 biți	mov BX, [EAX]	67:8B18	Adresare pe 32 biți prin DS:[EAX], deci se folosește prefixul 67h
16 biți	push dword [ebx]	66:67:FF33	Modul default este 16 biți; apare 67 pentru adresare pe 32 biți și 66 pentru operand DWORD
16 biți	push dword [CS:ebx]	2E:66:67:FF33	Prefix CS explicit (2E), operand size override (66) și address size override (67)
16 biți	rep push dword [CS:ebx]	F3:2E:66:67:FF33	REP prefix (superfluous pentru OllyDbg fix), urmat de prefixurile CS, operand size și address size
32 biți	mov eax, [bx]	67:8B07	Adresare pe 16 biți; 67 = address size override, rezultat în EAX (Offset_16_biti = [BX|BP] + [SI|DI] + [const])
16 biți	mov eax, [bx]	66:8B07	Operand size override (66) pentru a încărca în EAX, adresare pe 16 biți prin DS:[BX]

Reprezentarea instrucțiunilor mașină

- Octetul `ModR/M` (mod registru/memorie) specifică pentru unele dintre instrucţiuni natura şi locul operanzilor (registru, memorie).
- Acesta permite specificarea fie a unui registru, fie a unei locaţii de memorie a cărei adresă este exprimată prin intermediul unui offset *(link)*
- Dacă operandul exprimat este registru sau este un operand din memorie exprimat prin adresare directă sau adresare indirectă exprimată DOAR prin registrul de bază, octetul SIB NU mai apare. Deci în aceste 3 situaţii **octetul ModR/M apare SINGUR şi de sine stătător !** Apariţia sa în cadrul formatului intern al unei instucţiuni NU este condiţionată niciodată de octetul SIB ! Doar INVERS se pune problema.

Reprezentarea instrucțiunilor mașină

- Pentru cazuri mai complexe de adresare decât cele codificabile direct prin ModR/M, combinarea acestuia cu octetul SIB (Scale – Index – Base) permite următoarea formulă generală de definire a unui offset:

  `offset=[bază]+[index×scală]+[constantă]`

  (SIB) (deplasament + imediat)
- unde pentru bază şi index vor fi folosite valorile a doi regiştri, iar scală este 1, 2, 4 sau 8.
- Regiştrii permişi ca bază sau / şi index sunt: EAX, EBX, ECX, EDX, EBP, ESI, EDI.
- Registrul ESP este disponibil ca bază însă nu poate fi folosit cu rol de index. *(link)*
- Octetul SIB exprimă **primele 2 elemente din formula de calcul a offsetului unui operand (baza + index*scală). Apariţia octetului SIB este condiţionată de octetul ModR/M care exprimă prin conţinutul său un operand din memorie în care apar atât adresarea bazată cât şi cea indexată!**
- **Ca urmare, octetul SIB apare în şi codifică DOAR acele situaţii în care apar atât registrul de bază cât şi registrul de index în formula de calcul al offset-ului unui operand! Ca urmare octetul SIB NU poate apărea de sine stătător ci doar împreună cu MODR/M !**

Reprezentarea instrucțiunilor mașină (18. Reprezentarea instructiunilor masina - 04_05_Intel_MODRM_SIB_Tables.pdf)

Structura octetului ModR/M şi semnificaţia câmpurilor din această structură

7	6	5	4	3	2	1	0
MOD		REG / OPCODE			MOD R/M		

- ***Biţii 0-2 – Mod R/M – identifică codificarea operanzilor de tip:***
  i). adresare la memorie indirectă DOAR BAZATĂ
  ii). adresare la memorie indirectă BAZATĂ + deplasament pe 8 sau 32 biţi
  iii). REGISTRU

- 3 biti – sunt 8 valori posibile (000b-111b) prezentate pe COLOANA 3 - ALEGE PRIMUL OPERAND – cel din stânga, adica DESTINATIA

- ***Biţii 3-5 – Reg/Opcode – identifică operanzii de tip REGISTRU (deci dacă atât ModR/M cât şi Reg/Opcode exprimă regiştrii avem o instrucţiune de tip "instr reg, reg")***

- 3 biţi – sunt 8 valori posibile (000b-111b) prezentate pe LINIILE 6 şi 7 – din prima linie de sus a tabelului (REG=) – ALEGE AL DOILEA OPERAND - cel din dreapta, adică SURSA

- ***Biţii 6-7 - câmpul Mod - indică moduri de combinaţie de tipuri de operanzi:***
  00 – "memorie (doar bază), registru" sau "registru, memorie(doar bază)
  01 - "memorie (bază+disp8), registru" sau invers
  10 - "memorie (bază+disp32), registru" sau invers

Reprezentarea instrucțiunilor mașină (18. Reprezentarea instructiunilor masina - 04_05_Intel_MODRM_SIB_Tables.pdf)

Structura octetului SIB şi semnificaţia câmpurilor din această structură

7	6	5	4	3	2	1	0
SCALĂ		INDEX			BAZĂ		

- ***Biţii 0-2 – Bază – identifică registrul de bază (sunt 3 biţi = 8 valori posibile = 8 regişţri de bază posibili)***
- ***Biţii 3-5 – Index – identifică reg. de INDEX (3 biţi = 8 valori posibile ce exprimă 7 regişţri de bază posibili)***
- ***Biţii 6-7 – Scală – identifică scala (2 biţi = 4 valori posibile => 00 = 1, 01 =2, 10 = 4, 11 = 8)***

Reprezentarea instrucțiunilor mașină

Prefix	Cod	Mod R/M	SIB	Deplasament	Imediat
0, 1, 2, 3 sau 4	1, 2 sau 3	0 sau 1	0 sau 1	0, 1, 2 sau 4	0, 1, 2 sau 4

- Majoritatea instrucţiunilor folosesc pentru reprezentare fie numai campul de cod, fie cod urmat de ModR/M.
- Deplasament apare în cazul unor forme de adresare particulare (operanzi din memorie) şi urmează direct după ModR/M sau SIB, când SIB este prezent. Acest câmp poate fi codificat fie pe octet, fie pe cuvânt, fie pe dublu cuvant (32 biţi).
- Ca şi consecinţă a imposibilităţii prezenţei mai multor câmpuri de ModR/M, SIB şi deplasament într-o instrucţiune, arhitectura 80x86 NU permite codificarea a două adrese de memorie în aceeaşi instrucţiune.
- Valoare imediată oferă posibilitatea definirii unui operand ca fiind o constantă numerică pe 1, 2 sau 4 octeţi. Când este prezent, acest câmp apare întotdeauna la sfârşitul instrucţiunii.

Reprezentarea instrucțiunilor mașină

Exemplu

```
mov [ebp+a],ebx
mov [edx],ebp
mov [edx+17], ecx
mov ebx, esi
mov ecx, [ebx]
mov ecx, [esp + ebx*4]
mov ecx, [esp + ebx*4 + a - 7]
```

```
899D 00104000      MOV DWORD PTR SS:[EBP+401000],EBX
892A               MOV DWORD PTR DS:[EDX],EBP
894A 11            MOV DWORD PTR DS:[EDX+11],ECX
89F3               MOV EBX,ESI
8B0B               MOV ECX,DWORD PTR DS:[EBX]
8B0C9C             MOV ECX,DWORD PTR SS:[EBX*4+ESP]
8B8C9C F90F4000    MOV ECX,DWORD PTR SS:[EBX*4+ESP+400FF9]
```

Reprezentarea instrucțiunilo

Exemplu

a). `mov [ebp+a], ebx`
`; 89 9D 00104000`

18. Reprezentarea instructiunilor masina - 01_Intel x86 Assembler Instruction Set Opcode Table.pdf

18. Reprezentarea instructiunilor masina - 04_05_Intel_MODRM_SIB_Tables.pdf

- 9Dh = ?
- 89h = ?

Table 2-2. 32-Bit Addressing Forms with the ModR/M Byte

r8(/r) / r16(/r) / r32(/r) / mm(/r) / xmm(/r) / /digit (Opcode) / REG =			AL / AX / EAX / MM0 / XMM0 / 0 / 000	CL / CX / ECX / MM1 / XMM1 / 1 / 001	DL / DX / EDX / MM2 / XMM2 / 2 / 010	BL / BX / EBX / MM3 / XMM3 / 3 / 011	AH / SP / ESP / MM4 / XMM4 / 4 / 100	CH / BP / EBP / MM5 / XMM5 / 5 / 101	DH / SI / ESI / MM6 / XMM6 / 6 / 110	BH / DI / EDI / MM7 / XMM7 / 7 / 111
Effective Address	**Mod**	**R/M**	**Value of ModR/M Byte (in Hexadecimal)**							
[EAX]	00	000	00	08	10	18	20	28	30	38
[ECX]		001	01	09	11	19	21	29	31	39
[EDX]		010	02	0A	12	1A	22	2A	32	3A
[EBX]		011	03	0B	13	1B	23	2B	33	3B
[--][--][1]		100	04	0C	14	1C	24	2C	34	3C
disp32[2]		101	05	0D	15	1D	25	2D	35	3D
[ESI]		110	06	0E	16	1E	26	2E	36	3E
[EDI]		111	07	0F	17	1F	27	2F	37	3F
[EAX]+disp8[3]	01	000	40	48	50	58	60	68	70	78
[ECX]+disp8		001	41	49	51	59	61	69	71	79
[EDX]+disp8		010	42	4A	52	5A	62	6A	72	7A
[EBX]+disp8		011	43	4B	53	5B	63	6B	73	7B
[--][--]+disp8		100	44	4C	54	5C	64	6C	74	7C
[EBP]+disp8		101	45	4D	55	5D	65	6D	75	7D
[ESI]+disp8		110	46	4E	56	5E	66	6E	76	7E
[EDI]+disp8		111	47	4F	57	5F	67	6F	77	7F
[EAX]+disp32	10	000	80	88	90	98	A0	A8	B0	B8
[ECX]+disp32		001	81	89	91	99	A1	A9	B1	B9
[EDX]+disp32		010	82	8A	92	9A	A2	AA	B2	BA
[EBX]+disp32		011	83	8B	93	9B	A3	AB	B3	BB
[--][--]+disp32		100	84	8C	94	9C	A4	AC	B4	BC
[EBP]+disp32		101	85	8D	95	9D	A5	AD	B5	BD
[ESI]+disp32		110	86	8E	96	9E	A6	AE	B6	BE
[EDI]+disp32		111	87	8F	97	9F	A7	AF	B7	BF
EAX/AX/AL/MM0/XMM0	11	000	C0	C8	D0	D8	E0	E8	F0	F8
ECX/CX/CL/MM/XMM1		001	C1	C9	D1	D9	E1	E9	F1	F9
EDX/DX/DL/MM2/XMM2		010	C2	CA	D2	DA	E2	EA	F2	FA
EBX/BX/BL/MM3/XMM3		011	C3	CB	D3	DB	E3	EB	F3	FB
ESP/SP/AH/MM4/XMM4		100	C4	CC	D4	DC	E4	EC	F4	FC
EBP/BP/CH/MM5/XMM5		101	C5	CD	D5	DD	E5	ED	F5	FD
ESI/SI/DH/MM6/XMM6		110	C6	CE	D6	DE	E6	EE	F6	FE
EDI/DI/BH/MM7/XMM7		111	C7	CF	D7	DF	E7	EF	F7	FF

Reprezentarea instrucțiunilor mașină

Exemplu

a). `mov [ebp+a], ebx ; 899D 00104000`

18. Reprezentarea instructiunilor masina - 01_Intel x86 Assembler Instruction Set Opcode Table.pdf
18. Reprezentarea instructiunilor masina - 04_05_Intel_MODRM_SIB_Tables.pdf

`89h = MOV Ev,Gv`	`9Dh = 1001 1101b = 10 011 101b` (Mod R/M)		

	`Mod = 10`
	ma uit pe linia coresp. valorii 10
	`([...] + disp32)`
		`Reg/Opcode = 011`
		ma uit pe coloana coresp valorii 011 (`BL, BX, EBX, ...`)
			`r/M = 101`
			identifica linia `[EBP]+disp32` din sectiunea Mod=10

- Deci daca Mod R/M = 9Dh, setul de operanzi va fi: `instructiune [EBP]+disp32, EBX` adica o instructiune de tipul "instr memorie, registru"

18. Reprezentarea instructiunilor masina - 03_Addressing Method Codes.pdf

E - A ModR/M byte follows the opcode and specifies the operand. The operand is either a general-purpose register or a memory address. If it is a memory address, the address is computed from a segment register and any of the following values: a base register, an index register, a scaling factor, a displacement.

G - The reg field of the ModR/M byte selects a general register (for example, AX (000), EBX(011) etc.)

v - Word or doubleword, depending on operand-size attribute.

Reprezentarea instrucțiunilor mașină

Exemplu

b) `mov [edx],ebp ; 892A`

`89h = MOV Ev,Gv`

`2Ah - 0010 1010b = 00 101 010b` (Mod R/M)

- Deci `Mod = 00` - deci ma uit pe linia coresp. valorii 00 pt MOD ([...] - fara deplasamente) = [reg]
- `Reg/Opcode = 101` - gasesc valoarea 101 pe coloana din dreapta (CH, BP, EBP, ...)
- `r/M = 010` - identifica linia [`EDX`] din sectiunea `Mod = 00`
- Deci daca `Mod R/M = 2Ah`, setul de operanzi va fi: `instructiune [EDX], EBP` (sau `CH`, sau `BP`) - adica tot o instructiune de tipul "instr memorie, registru"

Reprezentarea instrucțiunilor mașină

Exemplu

c) `mov [edx+17], ecx ; 894A 11`

`89h = MOV Ev,Gv` (memory address, general register)

`4Ah = 0100 1010b = 01 001 010b` (Mod R/M)

- `Mod = 01` = [reg] + disp8
- `Reg/Opcode = 001` = ECX (operandul sursă – al DOILEA operand) [EDX + disp8]
- `r/M = 010` - identifică linia [EDX] + disp8 din sectiunea `Mod = 01` (acesta este PRIMUL OPERAND – DESTINATIA)
- In codificarea intructiunii mai apare in final 11h = 17 (in baza 10) – *“The disp8 nomenclature denotes an 8-bit displacement that follows ModR/M byte”*

Reprezentarea instrucțiunilor mașină

Exemplu

d) `mov ebx, esi ; 89 F3`

`89h = MOV Ev,Gv` (memory address or general register, general register)

`F3h = 1111 0011b = 11 110 011b (mod R/M)`

- Deci `Mod = 11` - deci mă uit pe linia coresp. valorii 11 pt MOD - aici este vorba despre un operand de tip REGISTRU si NU din memorie !
- `Reg/Opcode = 110` - gasesc valoarea 110 pe coloana din dreapta (DH, SI, ESI, ...)
- `r/M = 011` - identifica linia EBX/BX/BL din sectiunea Mod=11 (acesta este PRIMUL OPERAND - DESTINATIA)
- Deci daca `Mod R/M = F3h`, setul de operanzi va fi: instructiune EBX (sau BX sau BL), ESI (sau SI, sau DH) - adica o instructiune de tipul "registru, registru"

Reprezentarea instrucțiunilor mașină

Exemplu

e) `mov ecx, [ebx] ; 8B0B`

`8Bh = MOV Gv, Ev ; mov registru, memorie`

`0Bh = 00 001 011` (ModR/M)

- Deci `Mod = 00` - deci ma uit pe linia coresp. valorii 00 pt MOD ([...] - fara deplasamente)
- `Reg/Opcode = 001` - gasesc valoarea 001 pe coloana din dreapta (CL, CX, ECX, ...)

G - The reg field of the ModR/M byte selects a general register (for example, CX (001))

- `r/M = 011` - identifica linia [EBX] din sectiunea Mod=00 (acesta este AL DOILEA OPERAND - SURSA)

Reprezentarea instrucțiunilor mașină

Exemplu

f) `mov ecx, [esp + ebx*4] ; 8B0C9C`

`8Bh = MOV Gv, Ev` ; mov registru, memorie

`0Ch = 00 001 100b` (mod R/M byte)

- similar cu ceea ce este la e) , mai putin câmpul r/M:
- `r/M = 100` - identifica linia [EBX] din sectiunea Mod=00 (*1. The [--][--] nomenclature means a SIB follows the ModR/M byte*).

18. Reprezentarea instructiunilor masina - 04_05_Intel_MODRM_SIB_Tables.pdf
Table 2-3. 32-Bit Addressing Forms with the SIB Byte

- SIB_Byte Values - Structura lui este SS (SCALE - 2 biti) Index (3 biti) Base (3 biti)

`9Ch = 1001 1100 = 10 011 100` (SIB Byte)

- `SS = 10b` (factor de scalare = 4) - vezi coloana SS din tabel si lista de formule de adresare disponibila de pe prima coloana
- `Index = 011b` (vezi coloana 3 din linia coresp. Lui SS=10) deci e vorba despre expresia de index [EBX*4]
- `Base = 100b` (vezi linia 1) deci registrul de baza este ESP

Reprezentarea instrucțiunilor mașină

Exemplu

g) `mov ecx, [esp + ebx*4 + a - 7] ; 8B8C9C F90F4000`

`8Bh = MOV Gv, Ev` ; mov registru, memorie

`(mov ecx, dword ptr SS:[ebx*4 + esp + 00400FF9])`

`00401000h - 7 = 004000FF9h`

`8Ch = 10 001 100` (mod R/M byte)

- Deci `Mod = 10` - deci ma uit pe linia coresp. valorii 10 pt MOD ([...] + disp32)
- `Reg/Opcode = 001` - gasesc valoarea 001 pe coloana din dreapta (CL, CX, ECX, ...). Acesta este PRIMUL OPERAND (destinatia) = ECX
- `r/M = 100` - identifica din secţiunea Mod=10 (*1. The [--][--]+disp32 nomenclature means a SIB follows the ModR/M byte*).

9Ch = SIB byte

`F90F4000` = disp32 (= a-7 = 00400FF9h)

Concluzii

Pt Examenul Scris:

- Formatul intern al unei instrucţiuni este variabil, el putând ocupa între 1 şi 15 octeţi, având următoarea formă generală de reprezentare (Instructions byte-codes from OllyDbg):

```
[prefixe] + cod + [ModR/M] + [SIB] + [deplasament] + [imediat]
```

- Prefixele – material separat – 4 categorii (2 explicite, 2 implicite – 66h si 67h) + Clasificare si câteva exemple pregătite de acasă (diferite de ceea ce apare pe slides) care sa poată fi explicate în scris – câte 2 din fiecare categorie; Câte prefixe pot apărea simultan în cadrul unei instrucţiuni (exemplu !)
- Octet cod – obligatoriu (exemplu)
  - CE reprezintă şi ce rol au in reprezentarea unei instrucţiuni maşină octetii ModR/M şi SIB şi care este STRUCTURA LOR ? (câte UN exemplu de instrucţiune de tipul celor de la curs cu identificarea acestor octeţi ca structură si conţinut în formatul intern al acestora reflectat în OllyDbg)
  - Ce exprimă câmpurile [deplasament] şi [imediat]
  - Exemple de format intern care conţin aceste câmpuri cu explicaţii

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
UNIVERSITATEA BABEȘ-BOLYAI

Str. Mihail Kogălniceanu nr. 1
Cluj-Napoca, Cluj, România

www.cs.ubbcluj.ro