Intel 8088 Disasembleris

Kompiuterių architektūros praktinio darbo pristatymas

Tomas Giedraitis, Informatika, 3 kursas

Vilnius, 2021

Vilniaus Universitetas
Matematikos ir Informatikos fakultetas

Turinys

- Darbo tikslas
- Uždaviniai
- Iššūkiai darbo procese
- Atradimas: aštuntainiai kodai
- Darbo rezultatas
- Kompiuterių architektūros principai, pritaikyti praktiniame darbe

Darbo tikslas


[1]

- Intel 8088 16-bitų mikroprocesoriaus disasembleris
- Programavimo kalba: x86 Assembly, skirta x8086 16-bitų procesoriams (įskaitant Intel 8088)
- Asembliuojama su Turbo Assembler (TASM) programa. Atitinkamai pakinta ir kalbos sintaksė. Naudojama TASM direktyva “.8086”
- Galimybė apdoroti .COM ir .EXE failus
- Asembliavus disasembliuotą kodą (kodas pateiktas įvesties faile), rezultate turėtų gautis toks pat binarinis failas, kaip ir pirminis įvesties failas

Intel 8088, (1979, 1981)
Naudotas IBM PC, 1981 - 1987


[2]

Uždaviniai

```
0000 00dw mod reg r/m [poslinkis] – ADD registras += registras/atmintis
0000 010w bojb [bovb] – ADD akumuliatorius += betarpiškas operandas
000sr 110 – PUSH segmento registras
000sr 111 – POP segmento registras
0000 10dw mod reg r/m [poslinkis] – OR registras V registras/atmintis
0000 110w bojb [bovb] – OR akumuliatorius V betarpiškas operandas
0001 00dw mod reg r/m [poslinkis] – ADC registras += registras/atmintis
0001 010w bojb [bovb] – ADC akumuliatorius += betarpiškas operandas
0001 10dw mod reg r/m [poslinkis] – SBB registras -= registras/atmintis
0001 110w bojb [bovb] – SBB akumuliatorius -= betarpiškas operandas
```
[3]

- Disasembliavimas su rašikliu ant popieriaus (proceso suvokimas)
- Paieškos medžio sudarymas, kurio lapuose prieinama prie konkrečios instrukcijos (su atitinkamais argumentais)
- TASM specifinės sintaksės įsisavinimas kiekvienai komandai, kurią galima potencialiai atpažinti
- Dėsningumų aptikimas ir optimizacija jų pagrindu

Iššūkiai darbo procese

- Kiekvienos operacijos, atvaizduojamos mašininiu kodu, subtilumai ir jų įsisavinimas

  (136 skirtingos operacijos, 105 iš jų - su kintamais argumentais, 14 iš jų - kiek paprastesnės, nes argumentas nurodo tik arba kuris registras naudojamas, arba nurodoma, ar operacija dirbs su žodžiu ar su baitu).

- Kodo skaitomumas

- Kodo moduliarumas

  Makrosų, procedūrų naudojimas, taip pat - disasemblerio logikos atskyrimas nuo failo skaitymo (duomenų paėmimas ir paruošimas disasembleriui) ir rašymo į failą (rezultato duomenų reprezentacija) logikos.

- Kodo efektyvumas

  Susijęs ir su kodo skaitomumu. Pasirinkimas kiekvieną sykį, ar tam tikrai užduočiai atlikti bus naudojama atmintis, ar procesoriaus registrai.

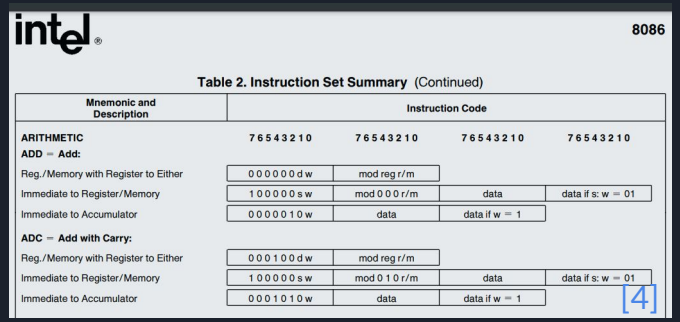
Iššūkiai darbo procese

- Koprocesoriaus Intel 8087 direktyvos (kviečiamos naudojant ESC kodus)
- MS-DOS (emuliuojama Dosbox programa) aplinkos įsisavinimas, komandos, trumpų Batch programėlių rašymas
- Skirtumai (ir jų suradimas bei patikrinimas) tarp operacijų palaikymo bei jų užrašymo būdo tarp:
  - oficialios Intel 8088 instrukcijų dokumentacijos
  - Turbo Assemblerio (TASM)
  - DEBUG.COM programos
  - Neoficialios vaizdinės Intel 8088 instrukcijų dokumentacijos lentelės pavidalu
  - Disasemblerio internete OnlineDisassembler.com, nustačius i8086 režimą

  Taip pat tarp Intel 8086 16 bitų procesoriaus ir jo vėlesnių įpėdinių (prisideda papildomos instrukcijos)

Iššūkiai darbo procese

Skirtumai (ir jų suradimas bei patikrinimas) tarp operacijų palaikymo bei jų užrašymo būdo:

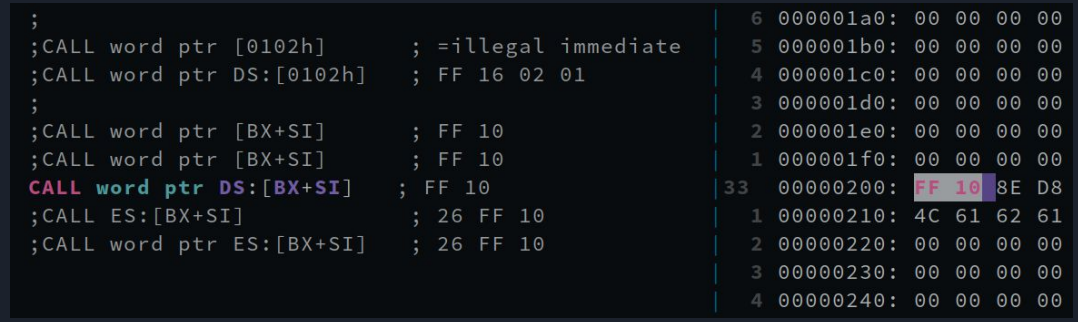


intel 8086

Table 2. Instruction Set Summary (Continued)

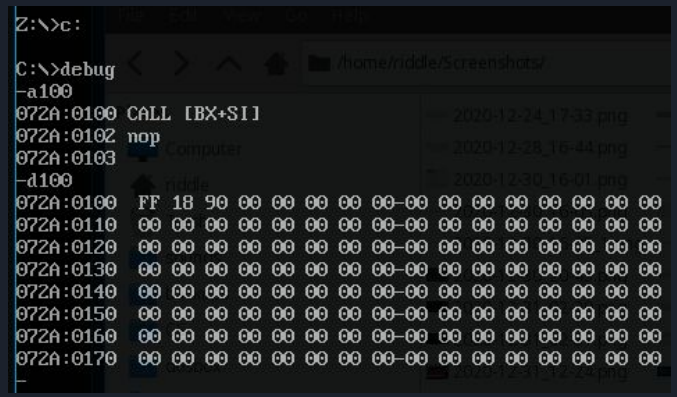
Mnemonic and Description	Instruction Code			
ARITHMETIC	76543210	76543210	76543210	76543210
ADD = Add:				
Reg./Memory with Register to Either	000000dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 000 r/m	data	data if s: w = 01
Immediate to Accumulator	0000010w	data	data if w = 1	
ADC = Add with Carry:				
Reg./Memory with Register to Either	000100dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 010 r/m	data	data if s: w = 01
Immediate to Accumulator	0001010w	data	data if w = 1	

[4]



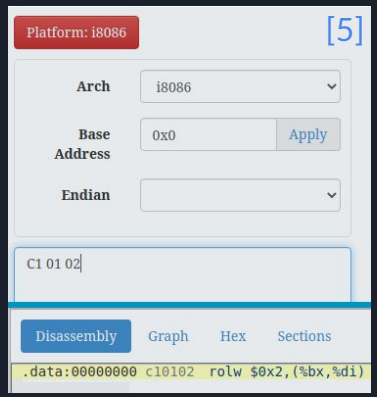
```
;
;CALL word ptr [0102h]       ; =illegal immediate
;CALL word ptr DS:[0102h]    ; FF 16 02 01
;
;CALL word ptr [BX+SI]       ; FF 10
;CALL word ptr [BX+SI]       ; FF 10
CALL word ptr DS:[BX+SI]    ; FF 10
;CALL ES:[BX+SI]             ; 26 FF 10
;CALL word ptr ES:[BX+SI]    ; 26 FF 10
```

```
  6 000001a0: 00 00 00 00
  5 000001b0: 00 00 00 00
  4 000001c0: 00 00 00 00
  3 000001d0: 00 00 00 00
  2 000001e0: 00 00 00 00
  1 000001f0: 00 00 00 00
33  00000200: FF 10 8E D8
  1 00000210: 4C 61 62 61
  2 00000220: 00 00 00 00
  3 00000230: 00 00 00 00
  4 00000240: 00 00 00 00
```



```
Z:\>c:

C:\>debug
-a100
072A:0100 CALL [BX+SI]
072A:0102 nop
072A:0103
-d100
072A:0100  FF 18 90 00 00 00 00 00-00 00 00 00 00 00 00 00
072A:0110  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
072A:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
072A:0130  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
072A:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
072A:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
072A:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
072A:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
-
```



Platform: i8086

[5]

Arch: i8086

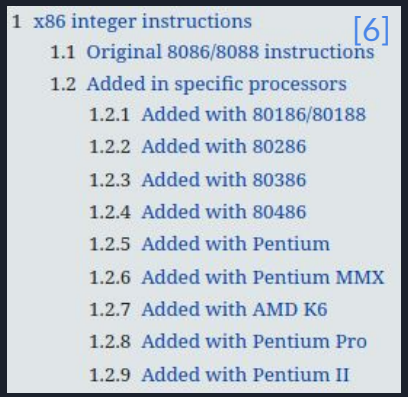
Base Address: 0x0 Apply

Endian:

C1 01 02

Disassembly | Graph | Hex | Sections

.data:00000000 c10102 rolw $0x2,(%bx,%di)



1 x86 integer instructions [6]

- 1.1 Original 8086/8088 instructions
- 1.2 Added in specific processors
  - 1.2.1 Added with 80186/80188
  - 1.2.2 Added with 80286
  - 1.2.3 Added with 80386
  - 1.2.4 Added with 80486
  - 1.2.5 Added with Pentium
  - 1.2.6 Added with Pentium MMX
  - 1.2.7 Added with AMD K6
  - 1.2.8 Added with Pentium Pro
  - 1.2.9 Added with Pentium II

Iššūkiai darbo procese

Skirtumai (ir jų suradimas bei patikrinimas) tarp operacijų palaikymo bei jų užrašymo būdo:

Intel 8088 instruction set [7]

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	ADD r/m8,r8 2+ 3/24+ 0---SZAPC	ADD r/m16,r16 2+ 3/24+ 0---SZAPC	ADD r8,r/m8 2+ 3/13+ 0---SZAPC	ADD r16,r/m16 2+ 3/13+ 0---SZAPC	ADD AL,d8 2 4 0---SZAPC	ADD AX,d16 3 4 0---SZAPC	PUSH ES 1 14 ---------	POP ES 1 12 ---------	OR r/m8,r8 2+ 3/24+ 0---SZAPC	OR r/m16,r16 2+ 3/24+ 0---SZAPC	OR r8,r/m8 2+ 3/13+ 0---SZAPC	OR r16,r/m16 2+ 3/13+ 0---SZAPC	OR AL,d8 2 4 0---SZAPC	OR AX,d16 3 4 0---SZAPC	PUSH CS 1 14 ---------	*POP CS 1 12 ---------
1x	ADC r/m8,r8 2+ 3/24+ 0---SZAPC	ADC r/m16,r16 2+ 3/24+ 0---SZAPC	ADC r8,r/m8 2+ 3/13+ 0---SZAPC	ADC r16,r/m16 2+ 3/13+ 0---SZAPC	ADC AL,d8 2 4 0---SZAPC	ADC AX,d16 3 4 0---SZAPC	PUSH SS 1 14 ---------	POP SS 1 12 ---------	SBB r/m8,r8 2+ 3/24+ 0---SZAPC	SBB r/m16,r16 2+ 3/24+ 0---SZAPC	SBB r8,r/m8 2+ 3/13+ 0---SZAPC	SBB r16,r/m16 2+ 3/13+ 0---SZAPC	SBB AL,d8 2 4 0---SZAPC	SBB AX,d16 3 4 0---SZAPC	PUSH DS 1 14 ---------	POP DS 1 12 ---------
2x	AND r/m8,r8 2+ 3/24+ 0---SZAPC	AND r/m16,r16 2+ 3/24+ 0---SZAPC	AND r8,r/m8 2+ 3/13+ 0---SZAPC	AND r16,r/m16 2+ 3/13+ 0---SZAPC	AND AL,d8 2 4 0---SZAPC	AND AX,d16 3 4 0---SZAPC	ES: 1 2 ---------	DAA 1 4 0---SZAPC	SUB r/m8,r8 2+ 3/24+ 0---SZAPC	SUB r/m16,r16 2+ 3/24+ 0---SZAPC	SUB r8,r/m8 2+ 3/13+ 0---SZAPC	SUB r16,r/m16 2+ 3/13+ 0---SZAPC	SUB AL,d8 2 4 0---SZAPC	SUB AX,d16 3 4 0---SZAPC	CS: 1 2 ---------	DAS 1 4 0---SZAPC
3x	XOR r/m8,r8 2+ 3/24+ 0---SZAPC	XOR r/m16,r16 2+ 3/24+ 0---SZAPC	XOR r8,r/m8 2+ 3/13+ 0---SZAPC	XOR r16,r/m16 2+ 3/13+ 0---SZAPC	XOR AL,d8 2 4 0---SZAPC	XOR AX,d16 3 4 0---SZAPC	SS: 1 2 ---------	AAA 1 8 0---SZAPC	CMP r/m8,r8 2+ 3/24+ 0---SZAPC	CMP r/m16,r16 2+ 3/24+ 0---SZAPC	CMP r8,r/m8 2+ 3/13+ 0---SZAPC	CMP r16,r/m16 2+ 3/13+ 0---SZAPC	CMP AL,d8 2 4 0---SZAPC	CMP AX,d16 3 4 0---SZAPC	DS: 1 2 ---------	AAS 1 8 0---SZAPC
4x	INC AX 1 3 0---SZAP-	INC CX 1 3 0---SZAP-	INC DX 1 3 0---SZAP-	INC BX 1 3 0---SZAP-	INC SP 1 3 0---SZAP-	INC BP 1 3 0---SZAP-	INC SI 1 3 0---SZAP-	INC DI 1 3 0---SZAP-	DEC AX 1 3 0---SZAP-	DEC CX 1 3 0---SZAP-	DEC DX 1 3 0---SZAP-	DEC BX 1 3 0---SZAP-	DEC SP 1 3 0---SZAP-	DEC BP 1 3 0---SZAP-	DEC SI 1 3 0---SZAP-	DEC DI 1 3 0---SZAP-
5x	PUSH AX 1 15 ---------	PUSH CX 1 15 ---------	PUSH DX 1 15 ---------	PUSH BX 1 15 ---------	PUSH SP 1 15 ---------	PUSH BP 1 15 ---------	PUSH SI 1 15 ---------	PUSH DI 1 15 ---------	POP AX 1 12 ---------	POP CX 1 12 ---------	POP DX 1 12 ---------	POP BX 1 12 ---------	POP SP 1 12 ---------	POP BP 1 12 ---------	POP SI 1 12 ---------	POP DI 1 12 ---------
6x																
7x	JO rel8 2 16/4 ---------	JNO rel8 2 16/4 ---------	JB/JNAE rel8 2 16/4 ---------	JNB/JAE rel8 2 16/4 ---------	JE/JZ rel8 2 16/4 ---------	JNE/JNZ rel8 2 16/4 ---------	JBE/JNA rel8 2 16/4 ---------	JNBE/JA rel8 2 16/4 ---------	JS rel8 2 16/4 ---------	JNS rel8 2 16/4 ---------	JP/JPE rel8 2 16/4 ---------	JNP/JPO rel8 2 16/4 ---------	JL/JNGE rel8 2 16/4 ---------	JNL/JGE rel8 2 16/4 ---------	JLE/JNG rel8 2 16/4 ---------	JNLE/JG rel8 2 16/4 ---------
8x	_ALU1 r/m8,d8 3+ 4/23+ 0---SZAPC	_ALU1 r/m16,d16 4+ 4/23+ 0---SZAPC	* ALU1 r/m8,d8 3+ 4/23+ 0---SZAPC	_ALU1 r/m16,d8 3+ 4/23+ 0---SZAPC	TEST r/m8,r8 2+ 3/13+ 0---SZAPC	TEST r/m16,r16 2+ 3/13+ 0---SZAPC	XCHG r8,r/m8 2+ 4/25+ ---------	XCHG r16,r/m16 2+ 4/25+ ---------	MOV r/m8,r8 2+ 2/13+ ---------	MOV r/m16,r16 2+ 2/13+ ---------	MOV r8,r/m8 2+ 2/12+ ---------	MOV r16,r/m16 2+ 2/12+ ---------	_MOV r/m16,sr 2+ 2/13+ ---------	LEA r16,r/m16 2+ 2+ ---------	_MOV sr,r/m16 2+ 2/12+ ---------	POP r/m16 2+ 12/25+ ---------
9x	NOP 1 3 ---------	XCHG AX,CX 1 3 ---------	XCHG AX,DX 1 3 ---------	XCHG AX,BX 1 3 ---------	XCHG AX,SP 1 3 ---------	XCHG AX,BP 1 3 ---------	XCHG AX,SI 1 3 ---------	XCHG AX,DI 1 3 ---------	CBW 1 2 ---------	CWD 1 5 ---------	CALL seg:a16 5 36 ---------	WAIT 1 4 ---------	PUSHF 1 14 ---------	POPF 1 12 ODITSZAPC	SAHF 1 4 ----SZAPC	LAHF 1 4 ---------
Ax	MOV AL,[addr] 3 14 ---------	MOV AX,[addr] 3 14 ---------	MOV [addr],AL 3 14 ---------	MOV [addr],AX 3 14 ---------	MOVSB 1 18/9+17n ---------	MOVSW 1 26/9+25n ---------	CMPSB 1 22/9+22n 0---SZAPC	CMPSW 1 30/9+30n 0---SZAPC	TEST AL,d8 2 4 0---SZAP-	TEST AX,d16 3 4 0---SZAP-	STOSB 1 11/9+10n ---------	STOSW 1 15/9+14n ---------	LODSB 1 12/9+13n ---------	LODSW 1 16/9+17n ---------	SCASB 1 15/9+15n 0---SZAPC	SCASW 1 19/9+19n 0---SZAPC
Bx	MOV AL,d8 2 4 ---------	MOV CL,d8 2 4 ---------	MOV DL,d8 2 4 ---------	MOV BL,d8 2 4 ---------	MOV AH,d8 2 4 ---------	MOV CH,d8 2 4 ---------	MOV DH,d8 2 4 ---------	MOV BH,d8 2 4 ---------	MOV AX,d16 3 4 ---------	MOV CX,d16 3 4 ---------	MOV DX,d16 3 4 ---------	MOV BX,d16 3 4 ---------	MOV SP,d16 3 4 ---------	MOV BP,d16 3 4 ---------	MOV SI,d16 3 4 ---------	MOV DI,d16 3 4 ---------
Cx			RET d16 3 24 ---------	RET 1 20 ---------	LES r16,m32 2+ 24+ ---------	LDS r16,m32 2+ 24+ ---------	_MOV r/m8,d8 3+ 4/14+ ---------	_MOV r/m16,d16 4+ 4/14+ ---------			RETF d16 3 34 ---------	RETF 1 33 ---------	INT 3 1 72 OD------	INT d8 2 71 OD------	INTO 1 73/4 OD------	IRET 1 44 ODITSZAPC
Dx	_ROT r/m8,1 2+ 2/23+ 0---????C	_ROT r/m16,1 2+ 2/23+ 0---????C	_ROT r/m8,CL 2+ 8/28++4n 0---????C	_ROT r/m16,CL 2+ 8/28++4n 0---????C	AAM *d8 2 83 0---SZAPC	AAD *d8 2 60 0---SZAPC	*SALC 1 2 ---------	XLAT 1 11 ---------	ESC 0 2+ 2/12+ ---------	ESC 1 2+ 2/12+ ---------	ESC 2 2+ 2/12+ ---------	ESC 3 2+ 2/12+ ---------	ESC 4 2+ 2/12+ ---------	ESC 5 2+ 2/12+ ---------	ESC 6 2+ 2/12+ ---------	ESC 7 2+ 2/12+ ---------
Ex	LOOPNZ/NE rel8 2 18/6 ---------	LOOPZ/E rel8 2 18/6 ---------	LOOP rel8 2 17/5 ---------	JCXZ rel8 2 18/6 ---------	IN AL,[d8] 2 14 ---------	IN AX,[d8] 2 14 ---------	OUT [d8],AL 2 14 ---------	OUT [d8],AX 2 14 ---------	CALL rel16 3 23 ---------	JMP rel16 3 15 ---------	JMP seg:a16 5 15 ---------	JMP rel8 2 15 ---------	IN AL,[DX] 1 12 ---------	IN AX,[DX] 1 12 ---------	OUT [DX],AL 1 12 ---------	OUT [DX],AX 1 12 ---------
Fx	LOCK 1 2 ---------		REPNE/REPNZ 1 0 ---------	REP/REPE/REPZ 1 0 ---------	HLT 1 2 ---------	CMC 1 2 --------C	_ALU2 r/m8,d8 2+ ? ?---?????	_ALU2 r/m16,d16 2+ ? ?---?????	CLC 1 2 --------C	STC 1 2 --------C	CLI 1 2 --I------	STI 1 2 --I------	CLD 1 2 -D------	STD 1 2 -D------	_MISC r/m8 2+ 3/23+ ?---????-	_MISC r/m16 2+ ? ?---????-

Iššūkiai darbo procese

Skirtumai (ir jų suradimas bei patikrinimas) tarp operacijų palaikymo bei jų užrašymo būdo:

Groups of instructions [8]

	00	08	10	18	20	28	30	38
80 _ALU1 r/m8,d8	ADD r/m8,d8 3+ 4/23+ O---SZAPC	OR r/m8,d8 3+ 4/23+ O---SZAPC	ADC r/m8,d8 3+ 4/23+ O---SZAPC	SBB r/m8,d8 3+ 4/23+ O---SZAPC	AND r/m8,d8 3+ 4/23+ O---SZAPC	SUB r/m8,d8 3+ 4/23+ O---SZAPC	XOR r/m8,d8 3+ 4/23+ O---SZAPC	CMP r/m8,d8 3+ 4/23+ O---SZAPC
81 _ALU1 r/m16,d16	ADD r/m16,d16 3+ 4/23+ O---SZAPC	OR r/m16,d16 3+ 4/23+ O---SZAPC	ADC r/m16,d16 3+ 4/23+ O---SZAPC	SBB r/m16,d16 3+ 4/23+ O---SZAPC	AND r/m16,d16 3+ 4/23+ O---SZAPC	SUB r/m16,d16 3+ 4/23+ O---SZAPC	XOR r/m16,d16 3+ 4/23+ O---SZAPC	CMP r/m16,d16 3+ 4/23+ O---SZAPC
82 *_ALU1 r/m8,d8	*ADD r/m8,d8 3+ 4/23+ O---SZAPC	*OR r/m8,d8 3+ 4/23+ O---SZAPC	*ADC r/m8,d8 3+ 4/23+ O---SZAPC	*SBB r/m8,d8 3+ 4/23+ O---SZAPC	*AND r/m8,d8 3+ 4/23+ O---SZAPC	*SUB r/m8,d8 3+ 4/23+ O---SZAPC	*XOR r/m8,d8 3+ 4/23+ O---SZAPC	*CMP r/m8,d8 3+ 4/23+ O---SZAPC
83 _ALU1 r/m16,d8	ADD r/m16,d8 3+ 4/23+ O---SZAPC	OR r/m16,d8 3+ 4/23+ O---SZAPC	ADC r/m16,d8 3+ 4/23+ O---SZAPC	SBB r/m16,d8 3+ 4/23+ O---SZAPC	AND r/m16,d8 3+ 4/23+ O---SZAPC	SUB r/m16,d8 3+ 4/23+ O---SZAPC	XOR r/m16,d8 3+ 4/23+ O---SZAPC	CMP r/m16,d8 3+ 4/23+ O---SZAPC
8C _MOV r/m16,sr	MOV r/m16,ES 2+ 2/13+ ---------	MOV r/m16,CS 2+ 2/13+ ---------	MOV r/m16,SS 2+ 2/13+ ---------	MOV r/m16,DS 2+ 2/13+ ---------	*MOV r/m16,ES 2+ 2/13+ ---------	*MOV r/m16,CS 2+ 2/13+ ---------	*MOV r/m16,SS 2+ 2/13+ ---------	*MOV r/m16,DS 2+ 2/13+ ---------
8E _MOV sr,r/m16	MOV ES,r/m16 2+ 2/12+ ---------	MOV CS,r/m16 2+ 2/12+ ---------	MOV SS,r/m16 2+ 2/12+ ---------	MOV DS,r/m16 2+ 2/12+ ---------	*MOV ES,r/m16 2+ 2/12+ ---------	*MOV CS,r/m16 2+ 2/12+ ---------	*MOV SS,r/m16 2+ 2/12+ ---------	*MOV DS,r/m16 2+ 2/12+ ---------
8F _POP r/m16	POP r/m16 2+ 12/25+ ---------							
C6 _MOV r/m8,d8	MOV r/m8,d8 3+ 4/14+ ---------	*MOV r/m8,d8 3+ 4/14+ ---------	*MOV r/m8,d8 3+ 4/14+ ---------	*MOV r/m8,d8 3+ 4/14+ ---------	*MOV r/m8,d8 3+ 4/14+ ---------	*MOV r/m8,d8 3+ 4/14+ ---------	*MOV r/m8,d8 3+ 4/14+ ---------	*MOV r/m8,d8 3+ 4/14+ ---------
C7 _MOV r/m16,d16	MOV r/m16,d16 4+ 4/14+ ---------	*MOV r/m16,d16 4+ 4/14+ ---------	*MOV r/m16,d16 4+ 4/14+ ---------	*MOV r/m16,d16 4+ 4/14+ ---------	*MOV r/m16,d16 4+ 4/14+ ---------	*MOV r/m16,d16 4+ 4/14+ ---------	*MOV r/m16,d16 4+ 4/14+ ---------	*MOV r/m16,d16 4+ 4/14+ ---------
D0 _ROT r/m8,1	ROL r/m8,1 2+ 2/23+ O-------C	ROR r/m8,1 2+ 2/23+ O-------C	RCL r/m8,1 2+ 2/23+ O-------C	RCR r/m8,1 2+ 2/23+ O-------C	SHL/SAL r/m8,1 2+ 2/23+ O---SZAPC	SHR r/m8,1 2+ 2/23+ O---SZAPC	*SHL/*SAL r/m8,1 2+ 2/23+ O---SZAPC	SAR r/m8,1 2+ 2/23+ O---SZAPC
D1 _ROT r/m16,1	ROL r/m16,1 2+ 2/23+ O-------C	ROR r/m16,1 2+ 2/23+ O-------C	RCL r/m16,1 2+ 2/23+ O-------C	RCR r/m16,1 2+ 2/23+ O-------C	SHL/SAL r/m16,1 2+ 2/23+ O---SZAPC	SHR r/m16,1 2+ 2/23+ O---SZAPC	*SHL/*SAL r/m16,1 2+ 2/23+ O---SZAPC	SAR r/m16,1 2+ 2/23+ O---SZAPC
D2 _ROT r/m8,CL	ROL r/m8,CL 2+ 8/28++4n O-------C	ROR r/m8,CL 2+ 8/28++4n O-------C	RCL r/m8,CL 2+ 8/28++4n O-------C	RCR r/m8,CL 2+ 8/28++4n O-------C	SHL/SAL r/m8,CL 2+ 8/28++4n O---SZAPC	SHR r/m8,CL 2+ 8/28++4n O---SZAPC	*SHL/*SAL r/m8,CL 2+ 8/28++4n O---SZAPC	SAR r/m8,CL 2+ 8/28++4n O---SZAPC
D3 _ROT r/m16,CL	ROL r/m16,CL 2+ 8/28++4n O-------C	ROR r/m16,CL 2+ 8/28++4n O-------C	RCL r/m16,CL 2+ 8/28++4n O-------C	RCR r/m16,CL 2+ 8/28++4n O-------C	SHL/SAL r/m16,CL 2+ 8/28++4n O---SZAPC	SHR r/m16,CL 2+ 8/28++4n O---SZAPC	*SHL/*SAL r/m16,CL 2+ 8/28++4n O---SZAPC	SAR r/m16,CL 2+ 8/28++4n O---SZAPC
F6 _ALU2 r/m8,d8	TEST r/m8,d8 3+ 5/11+ O---SZAPC	*TEST r/m8,d8 3+ 5/11+ O---SZAPC	NOT r/m8 2+ 3/24+ ---------	NEG r/m8 2+ 3/24+ O---SZAPC	MUL r/m8 2+ 70-77/76-83+ O---SZAPC	IMUL r/m8 2+ 80-98/86-104+ O---SZAPC	DIV r/m8 2+ 80-90/86-96+ O---SZAPC	IDIV r/m8 2+ 101-112/107-118+ O---SZAPC
F7 _ALU2 r/m16,d16	TEST r/m16,d16 3+ 5/11+ O---SZAPC	*TEST r/m16,d16 3+ 5/11+ O---SZAPC	NOT r/m16 2+ 3/24+ ---------	NEG r/m16 2+ 3/24+ O---SZAPC	MUL r/m16 2+ 118-133/124-139+ O---SZAPC	IMUL r/m16 2+ 128-154/134-160+ O---SZAPC	DIV r/m16 2+ 144-162/150-168+ O---SZAPC	IDIV r/m16 2+ 165-184/171-190+ O---SZAPC
FE _MISC r/m8	INC r/m8 2+ 3/23+ O---SZAP-	DEC r/m8 2+ 3/23+ O---SZAP-						
FF _MISC r/m16	INC r/m16 2+ 3/23+ O---SZAP-	DEC r/m16 2+ 3/23+ O---SZAP-	CALL r/m16 2+ 24/29+ ---------	CALL m32 2+ 53+ ---------	JMP r/m16 2+ 11/18+ ---------	JMP m32 2+ 24+ ---------	PUSH r/m16 2+ 15/24+ ---------	

Atradimas: aštuntainiai kodai

- Operacijos kodai paprastai skaitomi pabičiui disasemblerio kode (galima sakyti, skaitoma dvejetainiu pavidalu), o dokumentacijoje atvaizduojami dvejetainiu arba šešioliktainiu kodu.

- Pavertus kiekvieną instrukciją aštuntainiu kodu, paaiškėja tam tikra instrukcijų grupavimo logika, gali būti lengviau susidaryti paieškos medį, kodo skaitomumas gali pagerėti.

Pastaba: ši lentelė sudaryta iš 80x86 lentelės, pritaikius ją i8088, užtušavus kai kuriuos langelius, o kai kuriuos - paredagavus. Ji skirta tik orientacijai rašant disasemblerį. Nors instrukcijų pavadinimai sutampa, operandai (jų tipai) vietomis skiriasi nuo i8088

i8088 octal opcode tables

[9]

Pick a table 0xx-3xx, then down for the second octal digit, and across for the third digit.

0xx

	0	1	2	3	4	5	6	7
0	ADD r/m, reg		ADD reg, r/m		ADD acc, imm		PUSH ES	POP ES
1	OR r/m, reg		OR reg, r/m		OR acc, imm		CS	
2	ADC r/m, reg		ADC reg, r/m		ADC acc, imm		SS	SS
3	SBB r/m, reg		SBB reg, r/m		SBB acc, imm		DS	DS
4	AND r/m, reg		AND reg, r/m		AND acc, imm		ES:	DAA
5	SUB r/m, reg		SUB reg, r/m		SUB acc, imm		CS:	DAS
6	XOR r/m, reg		XOR reg, r/m		XOR acc, imm		SS:	AAA
7	CMP r/m, reg		CMP reg, r/m		CMP acc, imm		DS:	AAS

1xx

	0	1	2	3	4	5	6	7
0	INC reg eax	ecx	edx	ebx	esp	ebp	esi	edi
1	DEC reg eax	ecx	edx	ebx	esp	ebp	esi	edi
2	PUSH reg eax	ecx	edx	ebx	esp	ebp	esi	edi
3	POP reg eax	ecx	edx	ebx	esp	ebp	esi	edi
4								
5								
6	Jcc short O	NO	C	NC	E	NE	BE,NA	NBE,A
7	S	NS	P,PE	NP,PO	L,NGE	NL,GE	LE,NG	NLE,G

[10]

2xx

	0	1	2	3	4	5	6	7
0	*Group 1 (ALU)* r/m, imm			*Group 1* r/m, imm	TEST r/m, reg		XCHG r/m, reg	
1	MOV r/m, reg		MOV reg, r/m		MOV r/m, sreg	LEA reg, mem	MOV sreg, r/m	POP mem
2	NOP	XCHG reg, acc ecx	edx	ebx	esp	ebp	esi	edi
3	CBW	CWD, CWDE	CALL far	WAIT	PUSHF	POPF	SAHF	LAHF
4	MOV acc, disp		MOV disp, acc		MOVS B	W/D	CMPS B	W/D
5	TEST acc, imm		STOS B	W/D	LODS B	W/D	SCAS B	W/D
6	MOV reg, imm (byte) al	cl	dl	bl	ah	ch	dh	bh
7	MOV reg, imm (word/dword) eax	ecx	edx	ebx	esp	ebp	esi	edi

* Group 1: 0=ADD 1=OR 2=ADC 3=SBB 4=AND 5=SUB 6=XOR 7=CMP (ALU instructions)

3xx

	0	1	2	3	4	5	6	7
0			RET imm	RET	LES	LDS	MOV r/m, imm	
1			RETF imm	RETF	INT3	INT imm	INT0	IRET
2	*Group 2 (SHIFT)* r/m, 1		*Group 2 (SHIFT)* r/m, cl		AAM	AAD		XLAT
3	ESC (FPU instruction w/ "xrm" byte where x,r indicates instruction.)							
	0	1	2	3	4	5	6	7
4	LOOPNE short	LOOPE short	LOOP short	JCXZ short	IN acc, imm		OUT imm, acc	
5	CALL disp	JMP disp	JMP absolute	JMP short	IN acc, dx		OUT dx, acc	
6	LOCK		REPNE	REP REPE	HALT	CMC	*Group 3* byte r/m	word r/m
7	CLC	STC	CLI	STI	CLD	STD	*Group 4*	*Group 5*

* Group 2: 0=ROL 1=ROR 2=RCL 3=RCR 4=SHL 5=SHR 6=none 7=SAR
* Group 3: 0=TEST r/m, imm 1=none 2=NOT 3=NEG 4=MUL 5=IMUL 6=DIV 7=IDIV
* Group 4: 0=INC byte r/m 1=DEC byte r/m
* Group 5: 0 = INC word r/m; 1 = DEC word r/m; 2 = CALL near, absolute indirect; 3 = CALL far, absolute indirect; 4 = JMP far, absolute indirect; 5 = JMP far, absolute indirect; 6 = PUSH r/m; 7 = none

Kodo pavyzdys: aštuntainiai kodai

Testiniai duomenys

```
; ------------------------------------ GROUP 2 ------------------------------------------------
    db 2, 0, 0,  1, 0, 4,  2, 2, 2,  3, 3, 3                    ; 0???: ??      | ADD byte ptr [SI+377222], 333
    ;     80        44        92        DB                                      =ADD byte ptr [SI+92h], 0DBh
    ;     80        44        92        DB                                      =ADD byte ptr [SI-06Eh], 0DBh

    db 2, 0, 3,  2, 0, 4,  1, 1, 1,  2, 2, 2,  3, 3, 3          ; 0???: ??      | ADD word ptr [SI+222111], 377333
    ;     83        84        49        92        DB                          =ADD word ptr [SI+9249h], 0FFDBh
    ;                                                                         =ADD word ptr [SI-6DB7h], 0FFDBh

    db 2, 0, 0,  1, 1, 4,  2, 2, 2,  3, 3, 3                    ; 0???: ??      | OR byte ptr [SI+377222], 333
    db 2, 0, 3,  2, 2, 4,  1, 1, 1,  2, 2, 2,  3, 3, 3          ; 0???: ??      | ADC word ptr [SI+222111], 377333
```

Kodo pavyzdys: aštuntainiai kodai

Komandų atpažinimas

```
; ---------------------------------------------------------------
;  _20X
; ---------------------------------------------------------------
_20x:
    inc si
    mov al, byte ptr [data_octal+si]

    cmp al, 7
    ja undefined

    cmp al, 4
    jb short __20_0123

    cmp al, 6
    jb _20_45_test_reg_rm
    jmp _20_67_xchg_reg_rm

    __20_0123:
        inc si ; point to 'mod'
        inc si ; point SI to next octal digit after 'mod'
        mov bl, byte ptr [data_octal+si]
        dec si
        dec si ; return SI back
        ; find out which operation is used
        cmp bl, 4
        jb short __20_0123_mod_0123
        je _20_0123_and_rm_imm

        cmp bl, 6
        jb _20_0123_sub_rm_imm
        je _20_0123_xor_rm_imm
        jmp _20_0123_cmp_rm_imm

    __20_0123_mod_0123:
        cmp bl, 2
        jb short __20_0123_mod_01
        je _20_0123_adc_rm_imm
        jmp _20_0123_sbb_rm_imm

    __20_0123_mod_01:
        cmp bl, 1
        jb _20_0123_add_rm_imm
        jmp _20_0123_or_rm_imm
```

Darbo rezultatas šiuo metu:

Visų instrukcijų sėkmingas disasembliavimas.

Testiniai duomenys laikinai patalpinti pačios programos atmintyje.

Disasembliavus, gautas išeities failas sėkmingai vėl suasembliuojamas į mašininį kodą, kurį galima palyginti su testiniais duomenimis mašininiu pavidalu.

Šiuo metu darbo procese:

- Duomenų (mašininio kodo) skaitymas iš įvesties failo (.COM arba .EXE) ir rezultato (disasembliuotų instrukcijų) įrašymas į išeities failą (.ASM).
- Sėkmingas tokio išeities failo asembliavimas, pagaminant identišką mašininį kodą, koks buvo ir įvesties faile.
- Išeities faile prie kiekvienos atpažintos komandos taip pat nurodytas tos komandos pradžios adresas ir pačios komandos baitai šešioliktainiu pavidalu. Ši informacija turėtų būti užkomentuota, kad netrukdytų išeities failo asembliavimui.

Kompiuterių architektūros principai, pritaikyti praktiniame darbe

- Mašininių instrukcijų suvokimas kaip aukštesnio lygio programavimo kalbų kodo transliavimo rezultatas, efektyvumo klausimo gilesnis suvokimas per šią prizmę, ypač pagausėjus asemblerio kodui projekto pabaigoje. Kreipimosi į atmintį kaina.

- Mašininių instrukcijų suvokimas per jų įgyvendinimą mikroprocesoriuje, iki pat žemiausio - fizikinio (tranzistorių) lygmens.

- Įgūdis dirbti su baitais ir skaičiavimo sistemomis, praverčiantis ir bene visose kitose informatikos srityse. Kitaip tariant, mąstymas “mašinos kalba”.

Kompiuterių architektūros principai, pritaikyti praktiniame darbe

- Dėmesio kreipimo į detales svarba programavime ir šio įgūdžio lavinimas programuojant asembleriu.
- Bendros įžvalgos apie programavimo procesą, kilusios programuojant asembleriu.
- Darbo su abstrakcijomis lavinimas ir jo svarba, ypač atsižvelgiant į tai, kad Intel 8088 komandų sistema nėra ortogonali, o taip pat, kad kiekvienas mikroprocesorius turi savitas taisykles.

Šaltiniai

[1]. http://imgzoom.cdlib.org/Fullscreen.ics?ark=ark:/13030/kt9d5nc8p6/z1&&brand=oac4, Intel® 8088 Microprocessor Package, 1979. Creator: Intel Corporation

[2]. https://commons.wikimedia.org/wiki/File:IBM_PC-IMG_7271.jpg

[3]. https://klevas.mif.vu.lt/~julius/Tools/asm/KomKodaiViso.pdf

[4]. https://course.ece.cmu.edu/~ece740/f11/lib/exe/fetch.php?media=wiki:8086-datasheet.pdf

[5]. https://onlinedisassembler.com/odaweb/, selected platform: i8086

[6]. https://en.wikipedia.org/wiki/X86_instruction_listings

[7], [8]. https://pastraiser.com/cpu/i8088/i8088_opcodes.html, edited with GIMP program

[9],[10]. http://tom.bespin.org/src/low-level/opcodes.html, edited with GIMP program