LUCRAREA Nr. 3

INSTRUCȚIUNI DE PRELUCRARE A DATELOR ŞI INSTRUCȚIUNI DE CONTROL AL PROGRAMULUI (SALTURI PROPRIU-ZISE) PENTRU MICROPROCESOARELE COMPATIBILE INTEL x86 (IA-32) ÎN MODUL REAL

1. Scopul lucrării

Lucrarea de față îşi propune familiarizarea cu instrucțiunile de prelucrare a datelor precum şi cu o parte dintre instrucțiunile de control al programului (salturile propriu-zise, condiționate şi necondiționate) specifice microprocesoarelor compatibile Intel (IA-32) funcționând "în modul real".

2. Memoriu de instrucțiuni

Convențiile folosite sunt cele arătate în Lucrarea de laborator nr. 2. Se va acorda o atenție deosebită modului în care sunt afectate fanioanele. La convențiile amintite se adaugă:

nrcel: numărul de celule cu care se poate face deplasarea sau rotația unui operand.

2.1. Instrucțiuni aritmetice

		OF	DF	IF	TF	SF	ZF	AF	PF	CF
ADD d,s	Adunare	**x**				**x**	**x**	**x**	**x**	**x**

Descrierea formală a semanticii, în general: **(d) ← (d) + (s)**.

Operanzi	Exemple	Descrierea formală a semanticii
AL|AX, data	ADD AL,33H	**(AL) ← (AL) + 33H**
r, data	ADD CH,10H	**(CH) ← (CH) + 10H**
mem, data	ADD [BP],ALFA	**((SS)↑0H+(BP)) ← ((SS)↑0H+(BP)) + ALFA**
r_1, r_2	ADD CL,CH	**(CL) ← (CL) + (CH)**
r, mem	ADD SI,[SI+22H]	**(SI) ← (SI) + ((DS)↑0H + (SI) + 23H) ↑ ↑ ((DS)↑0H + (SI) + 22H)**
mem, r	ADD [BX],BX	**((DS)↑0H+(BX)+1) ↑ ((DS)↑0H+(BX)) ← ← ((DS)↑0H+(BX)+1) ↑ ((DS)↑0H+(BX)) + + (BX)**

ADC d,s	Adunare cu transport	OF DF IF TF SF ZF AF PF CF **x x x x x x**

Descrierea formală a semanticii, în general: **(d) ← (d) + (s) + (CF)**.

Operanzi	Exemple	**Descrierea formală a semanticii**
AL|AX, data	ADC AL,15H	**(AL) ← (AL) + 15H + (CF)**
r, data	ADC BX,0100H	**(BX) ← (BX) + 100H + (CF)**
mem, data	ADC [DI],1234H	**((DS)↑0H+(DI)+1) ↑ ((DS)↑0H+(DI)) ←** **← ((DS)↑0H+(DI)+1) ↑ ((DS)↑0H+(DI)) +** **+ 1234H +(CF)**
r_1, r_2	ADC AX,DI	**(AX) ← (AX) + (DI) + (CF)**
r, mem	ADC DX,[BX]	**(DX) ← (DX) +((DS)↑0H + (BX) + 1) ↑** **↑ ((DS)↑0H + (BX)) + (CF)**
mem, r	ADC [BX+DI+12H],DX	**((DS)↑0H + (BX) + (DI) + 13H) ↑** **↑ ((DS)↑0H + (BX) + (DI) +12H) ←** **← ((DS)↑0H + (BX) + (DI) + 13H) ↑** **↑ ((DS)↑0H + (BX) + (DI) +12H) +** **+ (DX) + (CF)**

INC d	Incrementarea destinației	OF DF IF TF SF ZF AF PF CF **x x x x x**

Descrierea formală a semanticii, în general: **(d) ← (d) + 1**.

Operanzi	Exemple	**Descrierea formală a semanticii**
r16	INC BX	**(BX) ← (BX) + 1**
r8	INC AL	**(AL) ← (AL) + 1**
mem	INC [BP+DI]	**((SS)↑0H + (BP) + (DI)) ←** **← ((SS)↑0H + (BP) + (DI)) + 1**

AAA	Ajustare ASCII pentru adunare	OF DF IF TF SF ZF AF PF CF **? ? ? x ? x**

Descrierea formală a semanticii: if **(AL) & 0F > 9** or **(AF) = 1** then
(AL) ← (AL) + 6
(AH) ← (AH) + 1
(AF) ← 1
(CF) ← (AF)
(AL) ← (AL) & 0F.

DAA	Ajustare zecimală pentru adunare	OF DF IF TF SF ZF AF PF CF **? x x x x x**

Descrierea formală a semanticii: if **(AL) & 0F > 9** or **(AF) = 1** then
(AL) ← (AL) + 06H
(AF) ← 1
if **(AL) > 9F** or **(CF) = 1** then
(AL) ← (AL) + 60H

SUB d,s	Scădere	OF DF IF TF SF ZF AF PF CF **x x x x x**

Descrierea formală a semanticii, în general: **(d) ← (d) - (s)**.

Operanzi	Exemple	**Descrierea formală a semanticii**
AL|AX, data	SUB AL,20H	**(AL) ← (AL) - 20H**
r, data	SUB BX,5566H	**(BX) ← (BX) - 5566H**
mem, data	SUB [BP+25H],444H	**((SS)↑0H+(BP)+26H) ↑** **↑ ((SS)↑0H+(BP)+25H) ←** **← ((SS)↑0H+(BP)+26H) ↑** **↑ ((SS)↑0H+(BP)+25H) -** **- 0444H**
r_1, r_2	SUB DX,DI	**(DX) ← (DX) - (DI)**
r, mem	SUB SI,[BX+100H]	**(SI) ← (SI) - ((DS)↑0H + (BX) + 101H) ↑** **↑ ((DS)↑0H + (BX) + 100H)**
mem, r	SUB [BP+50H],AX	**((SS)↑0H + (BP) + 51H) ↑** **↑ ((SS)↑0H + (BP) + 50H) ←** **← ((SS)↑0H + (BP) + 51H) ↑** **↑ ((SS)↑0H + (BP) + 50H) - (AX)**

SBB d,s	Scădere cu împrumut	OF DF IF TF SF ZF AF PF CF **x x x x x**

Descrierea formală a semanticii, în general: **(d) ← (d) - (s) - (CF)**.

Operanzi	Exemple	Descrierea formală a semanticii
AL|AX, data	SBB AX,1000H	**(AX) ← (AX) - 1000H - (CF)**
r, data	SBB DI,23H	**(DI) ← (DI) - 0023H - (CF)**
mem, data	SBB [BX+DI],33H	**((DS)↑0H+(BX)+(DI)) ←((DS)↑0H+(BX)+(DI))- - 33H - (CF)**
r_1, r_2	SBB AL,BL	**(AL) ← (AL) - (BL) - (CF)**
r, mem	SBB AH,[DI+55H]	**(AH) ← (AH) - ((DS)↑0H + (DI) + 55H) - (CF)**
mem, r	SBB [BX],DL	**((DS)↑0H + (BX)) ← ((DS)↑0H + (BX)) - - (DL) - (CF)**

		OF	DF	IF	TF	SF	ZF	AF	PF	CF
DEC d	Decrementarea destinației	**x**				**x**	**x**	**x**		**x**

Descrierea formală a semanticii, în general: **(d) ← (d) - 1**.

Operanzi	Exemple	Descrierea formală a semanticii
r16	DEC AX	**(AX) ← (AX) - 1**
r8	DEC DH	**(DH) ← (DH) - 1**
mem	DEC [BP+12H]	**((SS)↑0H + (BP) + 12H) ← ← ((SS)↑0H + (BP) + 12H) - 1**

		OF	DF	IF	TF	SF	ZF	AF	PF	CF
NEG d	Complementare față de 2 a destinației	**x**				**x**	**x**	**x**	**x**	**1***

Descrierea formală a semanticii, în general: **(d) ← 0H - (d)**.

Operanzi	Exemple	Descrierea formală a semanticii
r	NEG AX	**(AX) ← 0H - (AX)**
mem	NEG [DI]	**((DS)↑0H + (DI)) ← 0H - ((DS)↑0H + (DI))**

*** (CF) = 0** dacă **(d) = 0H**

		OF	DF	IF	TF	SF	ZF	AF	PF	CF
CMP s1,s2	Compararea prin scădere a doi operanzi	**x**				**x**	**x**	**x**	**x**	**x**

Descrierea formală a semanticii, în general: **(s_1) - (s_2)**.

Operanzi	Exemple	Descrierea formală a semanticii
AL|AX, data	CMP AX,0FFFFH	**(AX) - FFFFH**
r, data	CMP BX,10H	**(BX) - 0010H**
mem, data	CMP [BP+SI+5H],0ABH	**((SS)↑0H + (BP) + (SI) + 5H) - ABH**
r_1, r_2	CMP AL,CL	**(AL) - (CL)**
r, mem	CMP BH,[100H]	**(BH) - ((DS)↑0H + 100H)**
mem, r	CMP [BX+SI+45H],DX	**((DS)↑0H + (BX) + (SI) + 46H) ↑ ↑ ((DS)↑0H + (BX) + (SI) + 45H) - (DX)**

AAS	Ajustare ASCII pentru scădere	OF DF IF TF SF ZF AF PF CF **? ? ? x ? x**

Descrierea formală a semanticii: if **(AL) & 0F > 9** or **(AF) = 1** then
(AL) ← **(AL) - 6, (AH)** ← **(AH) - 1**
(AF) ← **1**
(CF) ← **(AF)**
(AL) ← **(AL) & 0F.**

DAS	Ajustare zecimală pentru scădere	OF DF IF TF SF ZF AF PF CF **? x x x x x**

Descrierea formală a semanticii: if **(AL) & 0F > 9** or **(AF) = 1** then
(AL) ← **(AL) - 06H**
(AF) ← **1**
if **(AL) > 9F** or **(CF) = 1** then
(AL) ← **(AL) - 60H**
(CF) ← **1**

MUL s	Înmulţire	OF DF IF TF SF ZF AF PF CF **x ? ? ? ? x**

Descrierea formală a semanticii, în general:
pentru operaţia pe 8 biţi:

(AX) ← **(AL) * (s)** if **(AH) = 0** then
(CF) ← **0**
else **(CF)** ← **1**
(OF) ← **(CF)** ,

iar pentru operaţia pe 16 biţi:

(DX)↑**(AX)** ← **(AX) * (s)** if **(DX) = 0** then
(CF) ← **0**
else **(CF)** ← **1**
(OF) ← **(CF)**.

Operanzi	Exemple	**Descrierea formală a semanticii**
r8	MUL DL	**(AX)** ← **(AL) * (DL)**
r16	MUL BX	**(DX)**↑**(AX)** ← **(AX) * (BX)**
mem8	MUL [BP+DI]	**(AX)** ← **(AL) * ((SS)**↑**0H + (BP) + (DI))**
mem16	MUL [1268H]	**(DX)**↑**(AX)** ← **(AX) * ((DS)**↑**0H + 1269H)** ↑ ↑ **((DS)**↑**0H + 1268H)**

IMUL s	Înmulţire cu semn	OF DF IF TF SF ZF AF PF CF **x ? ? ? ? x**

Descrierea formală a semanticii, în general:

- pentru operația pe 8 biti:

(AX) ← (AL) * (s)

if **(AH) = 0** or **(AH) = FF** then
(CF) ← 0
else **(CF) ← 1**
(OF) ← (CF) ,

- iar pentru operația pe 16 biți:

(DX)↑(AX) ← (AX) * (s)

if **(DX) = 0** or **(DX) = FFFF** then
(CF) ← 0
else **(CF) ← 1**
(OF) ← (CF).

Operanzi	Exemple	**Descrierea formală a semanticii**
r8	IMUL DL	**(AX) ← (AL) * (DL)**
r16	IMUL CX	**(DX)↑(AX) ← (AX) * (CX)**
mem8	IMUL [0ABCDH]	**(AX) ← (AL) * ((DS)↑0H + ABCDH)**
mem16	IMUL [BX+550H]	**(DX)↑(AX) ← (AX) * ((DS)↑0H+(BX)+551H) ↑ ↑ ((DS)↑0H+(BX)+550H)**

		OF	DF	IF	TF	SF	ZF	AF	PF	CF
AAM	Ajustare ASCII pentru înmulțire (după înmuțire)	**?**				**x**	**x**	**?**	**x**	**?**

Descrierea formală a semanticii: **(AH) ← (AL) div 0A**
(AL) ← (AL) mod 0A .

		OF	DF	IF	TF	SF	ZF	AF	PF	CF
DIV s	Împărțire	**?**				**?**	**?**	**?**	**?**	**?**

Descrierea formală a semanticii, în general:
pentru operația pe 8 biți:

if **(AX) div (s) > FF** then
(SP) ← (SP) - 2
((SS)↑0H + (SP) + 1) ↑ ((SS)↑0H + (SP)) ← (F)
(IF) ← 0
(TF) ← 0
(SP) ← (SP) - 2
((SS)↑0H + (SP) + 1) ↑ ((SS)↑0H + (SP)) ← (CS)
(CS) ← (00003H) ↑ (00002H)
(SP) ← (SP) - 2
((SS)↑0H + (SP) + 1) ↑ ((SS)↑0H + (SP)) ← (IP)
(IP) ← (00001H) ↑ (00000H)

else **(AL) ← (AX) div (s)**
(AH) ← (AX) mod (s) ,

iar pentru operația pe 16 biți:

if **(DX)↑(AX) div (s) > FFFF** then
(SP) ← (SP) - 2
((SS)↑0H + (SP) + 1) ↑ ((SS)↑0H + (SP)) ← (F)
(IF) ← 0
(TF) ← 0
(SP) ← (SP) - 2
((SS)↑0H + (SP) + 1) ↑ ((SS)↑0H + (SP)) ← (CS)
(CS) ← (00003H) ↑ (00002H)
(SP) ← (SP) - 2
((SS)↑0H + (SP) + 1) ↑ ((SS)↑0H + (SP)) ← (IP)
(IP) ← (00001H) ↑ (00000H)
else **(AX) ← (DX)↑(AX) div (s)**
(DX) ← (DX)↑(AX) mod (s) .

Operanzi	Exemple	**Descrierea formală a semanticii**
r8	DIV CL	**(AL) ← (AX) div (CL)** **(AH) ← (AX) mod (CL)**
r16	DIV BX	**(AX) ← (DX)↑(AX) div (BX)** **(DX) ← (DX)↑(AX) mod (BX)**
mem8	DIV [BP+50H]	**(AL) ← (AX) div ((SS)↑0H + (BP) + 50H)** **(AH) ← (AX) mod ((SS)↑0H + (BP) + 50H)**
mem16	DIV [DI+41H]	**(AX) ← (DX)↑(AX) div ((DS)↑0H+(DI)+42H) ↑ ↑ ((DS)↑0H+(DI)+41H)** **(DX)←(DX)↑(AX) mod ((DS)↑0H+(DI)+42H) ↑ ↑ ((DS)↑0H+(DI)+41H)**

		OF DF IF TF SF ZF AF PF CF
IDIV s	Împărțire cu semn	**? ? ? ? ? ?**

Descrierea formală a semanticii este similară cu cea de la instrucțiunea precedentă, cu excepția comparației inițiale care, pentru întregii cu semn, este:
pentru împărțirea pe 8 biți:

if **(AX) div (s) > 0** and **(AX) div (s) > FF** or
(AX) div (s) < 0 and **(AX) div (s) < 0-FF-1** then ... ,

iar pentru împărțirea pe 16 biți:

if **(DX)↑(AX) div (s) > 0** and **(DX)↑(AX) div (s) > FFFF** or
(DX)↑(AX) div (s) < 0 and **(DX)↑(AX) div (s) < 0-FFFF-1** then ...

Operanzi	Exemple	Descrierea formală a semanticii
r8	DIV BL	**(AL) ← (AX) div (BL)** **(AH) ← (AX) mod (BL)**
r16	DIV CX	**(AX) ← (DX)↑(AX) div (CX)** **(DX) ← (DX)↑(AX) mod (CX)**
mem8	DIV [BX+SI]	**(AL) ← (AX) div ((DS)↑0H + (BX) + (SI))** **(AH) ← (AX) mod ((DS)↑0H + (BX) + (SI))**
mem16	DIV [BP]	**(AX) ← (DX)↑(AX) div ((SS)↑0H +(BP) +1) ↑ ↑ ((SS)↑0H +(BP))** **(DX) ← (DX)↑(AX) mod ((SS)↑0H +(BP) +1)↑ ↑ ((SS)↑0H +(BP))**

		OF	DF	IF	TF	SF	ZF	AF	PF	CF
AAD	Ajustare ASCII pt. împărţire (înainte de împărţire)	**?**				**x**	**x**	**?**	**x**	**?**

Descrierea formală a semanticii: **(AL) ← (AH) * 0AH + (AL)**
(AH) ← 0H .

		OF	DF	IF	TF	SF	ZF	AF	PF	CF
CBW	Extindere (cu semn) a unui octet la un cuvânt									

Descrierea formală a semanticii: if **(AL) < 80H** then
(AH) ← 00H
else **(AH) ← FFH**.

		OF	DF	IF	TF	SF	ZF	AF	PF	CF
CWD	Extindere (cu semn) a unui cuvânt la un cuvânt dublu									

Descrierea formală a semanticii: if **(AX) < 8000H** then
(DX) ← 0000H
else **(DX) ← FFFFH**.

2.2. Operaţii logice

		OF	DF	IF	TF	SF	ZF	AF	PF	CF
NOT d	Complementare faţă de 1 a destinaţiei									

Descrierea formală a semanticii, în general: **(d) ← FFH - (d)** ,
pentru operand pe 8 biţi,
sau **(d) ← FFFFH - (d)**

pentru operand pe 16 biţi.

Operanzi	Exemple	**Descrierea formală a semanticii**
r	NOT AX	**(AX) ← FFFFH - (AX)**
mem	NOT [0EEFFH]	**((DS)↑0H +EEFFH) ← FFH - - ((DS)↑0H+EEFFH)**

		OF	DF	IF	TF	SF	ZF	AF	PF	CF
AND d,s	ŞI logic	**0**				**x**	**x**	**?**	**x**	**0**

Descrierea formală a semanticii, în general: **(d) ← (d) & (s)** .

Operanzi	Exemple	**Descrierea formală a semanticii**
AL|AX, data	AND AX,0FFH	**(AX) ← (AX) & 0033H**
r, data	AND CX,10H	**(CX) ← (CX) & 0010H**
mem, data	AND [DI],0AAAAH	**((DS)↑0H +(DI) +1) ↑ ((DS)↑0H +(DI)) ← ← ((DS)↑0H +(DI) +1) ↑ ((DS)↑0H +(DI)) & & AAAAH**
r_1, r_2	AND CL,DL	**(CL) ← (CL) & (DL)**
r, mem	AND DX,[BP]	**(DX) ← (DX) & ((SS)↑0H + (BP) + 1H) ↑ ↑ ((SS)↑0H + (BP))**
mem, r	AND [BX+DI],CL	**((DS)↑0H+(BX)+(DI)) ← ← ((DS)↑0H+(BX)+(DI)) & (CL)**

		OF	DF	IF	TF	SF	ZF	AF	PF	CF
TEST s1, s2	Compararea prin ŞI logic nedistructiv	**0**				**x**	**x**	**?**	**x**	**0**

Descrierea formală a semanticii, în general: **(s_1) & (s_2)** .

Operanzi	Exemple	**Descrierea formală a semanticii**
AL|AX, data	TEST AL,55H	**(AL) & 55H**
r, data	TEST DI,1234H	**(DI) & 1234H**
mem, data	TEST [SI],00101100B	**((DS)↑0H + (SI)) & 00101100B**
r_1, r_2	TEST DI,BX	**(DI) & (BX)**
r, mem	TEST CL,[SI]	**(CL) & ((DS)↑0H + (SI))**

		OF	DF	IF	TF	SF	ZF	AF	PF	CF
OR d,s	SAU logic	**0**				**x**	**x**	**?**	**x**	**0**

Descrierea formală a semanticii, în general: **(d) ← (d) ∀ (s)** .

Operanzi	Exemple	Descrierea formală a semanticii
AL|AX, data	OR AL,22H	**(AL) ← (AL) ∀ 22H**
r, data	OR DX,1FFFH	**(DX) ← (DX) ∀ 1FFFH**
mem, data	OR [BP+SI],1	**((SS)↑0H +(BP) + (SI)) ← ← ((SS)↑0H +(BP) + (SI)) ∀ 01H**
r_1, r_2	OR CL,BL	**(CL) ← (CL) ∀ (BL)**
r, mem	OR BX,[SI]	**(BX) ← (BX) ∀ ((DS)↑0H + (SI) + 1H) ↑ ↑ ((DS)↑0H + (SI))**
mem, r	OR [BP+DI],CX	**((SS)↑0H +(BP) +(DI) +1) ↑ ↑ ((SS)↑0H +(BP) +(DI)) ← ← ((SS)↑0H +(BP) +(DI) +1) ↑ ↑ ((SS)↑0H +(BP) +(DI)) ∀ ∀ (CX)**

		OF DF IF TF SF ZF AF PF CF
XOR d,s	SAU exclusiv	**0 x x ? x 0**

Descrierea formală a semanticii, în general: **(d) ← (d) ⊕ (s)** .

Operanzi	Exemple	Descrierea formală a semanticii
AL|AX, data	XOR AX,333H	**(AX) ← (AX) ⊕ 0333H**
r, data	XOR BP,245H	**(BP) ← (BP) ⊕ 0245H**
mem, data	XOR [DI],7788H	**((DS)↑0H +(DI) +1H) ↑ ((DS)↑0H +(DI)) ← ← ((DS)↑0H +(DI) +1H) ↑ ((DS)↑0H +(DI)) ⊕ ⊕ 7788H**
r_1, r_2	XOR DX,SI	**(DX) ← (DX) ⊕ (SI)**
r, mem	XOR CX,[BX+SI]	**(CX) ← (CX) ⊕ ((DS)↑0H +(BX) +(SI) +1H) ↑ ↑ ((DS)↑0H +(BX) +(SI))**
mem, r	XOR [DI+0AAH],BL	**((DS)↑0H +(DI) +AAH) ← ← ((DS)↑0H +(DI) +AAH) ⊕ (BL)**

2.3. Deplasări şi rotații

		OF DF IF TF SF ZF AF PF CF
SAL | SHL s,nrcel	Deplasare stânga logică sau aritmetică	**x x x ? x x**

Descrierea formală a semanticii:

while **nrcel ≠ 0** do
  (CF) ← $(s)_{msb}$
  (s) ← (s) * 2
  $(s)_{lsb}$ ← 0
  nrcel ← nrcel - 1
if **nrcel = 1** then
  if **$(s)_{msb}$ ≠ (CF)** then
    (OF) ← 1
  else **(OF) ← 0**
else **(OF)** nedeterminat .

Operanzi	Exemple
r, 1	SHL BX,1
r, CL	SAL DX,CL
mem, 1	SHL [BX+SI],1
mem, CL	SHL [DI+10H],CL

SHR s, nrcel	Deplasare dreapta logică	OF DF IF TF SF ZF AF PF CF **x x x ? x x**

Descrierea formală a semanticii:

while **nrcel ≠ 0** do
  (s)msb ← 0
  (s) ← (s) div 2
  (CF) ← (s)lsb
  nrcel ← nrcel - 1
if **nrcel = 1** then
  if **(s)msb ≠ (s)msb-1** then
    (OF) ← 1
  else **(OF) ← 0**
else **(OF)** nedeterminat .

Operanzi	Exemple
r, 1	SHR DL,1
r, CL	SHR BX,CL
mem, 1	SHR [DI],1
mem, CL	SHR [BP+SI+4H],CL

SAR s, nrcel	Deplasare dreapta aritmetică	OF DF IF TF SF ZF AF PF CF **x x x ? x x**

Descrierea formală a semanticii este similară cu cea a instrucțiunii precedente (singura deosebire fiind că **msb** trebuie să fie păstrat, iar **OF** este resetat şi dacă **nrcel ≠ 1**).

Operanzi	Exemple
r, 1	SAR BH,1
r, CL	SAR AX,CL
mem, 1	SAR [BX],1
mem, CL	SAR [BP+SI],CL

ROL s, nrcel	Rotație stânga	OF DF IF TF SF ZF AF PF CF **x x**

Descrierea formală a semanticii:

while **nrcel ≠ 0** do
  (s) ← (s) * 2
  (CF) ← (s)msb
  (s)lsb ← (s)msb
  nrcel ← nrcel - 1
if **nrcel = 1** then
  if **(s)msb ≠ (CF)** then
    (OF) ← 1
  else **(OF) ← 0**
else **(OF)** nedeterminat .

Operanzi	Exemple
r, 1	ROL SI,1
r, CL	ROL DX,CL
mem, 1	ROL [BX+DI],1
mem, CL	ROL [BP+100H],CL

ROR s, nrcel	Rotație dreapta	OF DF IF TF SF ZF AF PF CF **x** **x**

Descrierea formală a semanticii:

while **nrcel ≠ 0** do
  (s) ← (s) div 2
  $(s)_{msb}$ ← $(s)_{lsb}$
  (CF) ← $(s)_{lsb}$
  nrcel ← nrcel - 1
if **nrcel = 1** then
  if **$(s)_{msb}$ ≠ $(s)_{msb-1}$** then
    (OF) ← 1
  else **(OF) ← 0**
else **(OF)** nedeterminat .

Operanzi	Exemple
r, 1	ROR AX,1
r, CL	ROR DX,CL
mem, 1	ROR [BP],1
mem, CL	ROR [1000H],CL

RCL s, nrcel	Rotație stânga cu transport	OF DF IF TF SF ZF AF PF CF **x** **x**

Descrierea formală a semanticii este asemănătoare cu rotația stânga simplă.

Operanzi	Exemple
r, 1	RCL CX,1
r, CL	RCL AL,CL
mem, 1	RCL [SI],1
mem, CL	RCL [BX+DI],CL

RCR s,nrcel	Rotație dreapta cu transport	OF DF IF TF SF ZF AF PF CF **x** **x**

Descrierea formală a semanticii este asemănătoare cu a celorlalte rotații.

Operanzi	Exemple
r, 1	RCR AX,1
r, CL	RCR BX,CL
mem, 1	RCR [BP+DI],1
mem, CL	RCR [BX],CL

2.4. Salturi propriu-zise necondiționate

JMP adr	Salt propriu-zis, neconditionat	OF DF IF TF SF ZF AF PF CF

Descrierea formală a semanticii, în funcție de modul de adresare folosit:

a) Salt cu adresare absolută (directă) **intersegment**:

JMP adr32 ; **(CS) ← adr32$_h$**
(IP) ← adr32$_l$.

Sau, punând în evidență faptul că adresa completă face parte din formatul instrucțiunii:

(CS) ← ((CS)↑0H+(IP)+4) ↑ ((CS)↑0H+(IP)+3)
(IP) ← ((CS)↑0H+(IP)+2) ↑ ((CS)↑0H+(IP)+1).

b) Salt cu adresare relativă:

JMP disp8|disp16 ; **(IP) ← (IP) + disp8|disp16**

Deplasamentul face parte din formatul instrucțiunii curente.

c) Salt cu adresare în registru sau indirectă in memorie, **intrasegment**:

JMP r16 | mem16 ; **(IP) ← (r16) | (mem16)** .

d) Salt cu adresare indirectă în memorie, **intersegment**:

JMP mem32 ; **(CS) ← (mem32)$_h$**
(IP) ← (mem32)$_l$.

Operanzi	Exemple	Descrierea formală a semanticii
adr32	JMP ET-IN-ALT-SEG	**(CS)← ((CS)↑0H+(IP)+4) ↑ ((CS)↑0H+(IP)+3)** **(IP) ← ((CS)↑0H+(IP)+2) ↑ ((CS)↑0H+(IP)+1)**
disp16	JMP ET-IN-SEG	**(IP) ← (IP) +** **+ ((CS)↑0H+(IP)+2) ↑ ((CS)↑0H+(IP)+1)**
disp8	JMP FOARTE-APROAPE	**(IP) ← (IP) + ((CS)↑0H+(IP)+1)**
r16	JMP BX	**(IP) ← (BX)**
mem*	JMP [BX]	**(IP) ← ((DS)↑0H+(BX)+1) ↑ ((DS)↑0H+(BX))**
mem**	JMP [DI]	**(CS)← ((DS)↑0H+(DI)+3) ↑ ((DS)↑0H+(DI)+2)** **(IP) ← ((DS)↑0H+(DI)+1) ↑ ((DS)↑0H+(DI))**

* salt cu adresare indirectă definit cu directivă de asamblare ca salt intra-segment;
** salt cu adresare indirectă definit cu directivă de asamblare ca salt inter-segment.

2.5. Salturi condiționate

Jxx disp8	Salt propriu-zis, neconditionat	OF DF IF TF SF ZF AF PF CF

Descrierea formală a semanticii:

if **condiție** then
(IP) ← (IP) + disp8 , sau, detaliat:
(IP) ← (IP) + ((CS)↑0H+(IP)+1)

Adunarea se face prin extensie cu semn la un număr de16 biți.

Mnemonica	Cond. testată	Interpretare
JA | JNBE	(CF)sau(ZF)=0	Salt dacă "peste" | dacă "nu sub sau egal"
JAE|JNB|JNC	(CF)=0	Salt dacă "peste sau egal"|dacă "nu sub"|dacă "nu există transport"
JB|JNAE|JC	(CF)=1	Salt dacă "sub" | dacă "nu peste sau egal" | dacă "există transport"
JBE | JNA	(CF)sau(ZF)=1	Salt dacă "sub sau egal"| dacă "nu peste"
JE | JZ	(ZF)=1	Salt dacă "egal" | dacă " zero"
JG | JNLE	((SF)⊗(OF))sau(ZF)=0	Salt dacă "mai mare"| dacă "nu mai mic sau egal"
JGE | JNL	(SF)⊗(OF)=0	Salt dacă "mai mare sau egal "| dacă "nu mai mic"
JL | JNGE	(SF)⊗(OF)=1	Salt dacă "mai mic" | dacă "nu mai mare sau egal"
JLE | JNG	((SF)⊗(OF))sau(ZF)=1	Salt dacă "mai mic sau egal "| dacă "nu mai mare "
JNE | JNZ	(ZF)=0	Salt dacă "ne-egal" | dacă "non-zero"
JNO	(OF)=0	Salt dacă "nu există depăşire"
JNP | JPO	(PF)=0	Salt dacă " non- paritate"| dacă "impar"
JNS	(SF)=0	Salt dacă "non-semn" | dacă "pozitiv"
JO	(OF)=1	Salt dacă "există depaşire"
JP | JPE	(PF)=1	Salt dacă "există paritate"| dacă "par"
JS	(SF)=1	Salt dacă "exist semn"

NOTĂ: - condițiile care se traduc cu "*mai mare*" sau "*mai mic*" se referă la operații asupra unor **numere cu semn**;

- exprimările "*sub*" sau "*peste*" se aplică operațiilor asupra unor **numere fără semn**.

3. Indicații importante cu privire la operațiile aritmetice

Microprocesorul este un automat care execută anumite operații predefinite. Acesta nu poate sa cunoască semnificația operatorilor şi rezultatelor; de aceea, dacă programatorul nu controlează foarte strict aceste semnificații, este foarte probabil ca rezultatele obținute să fie eronate. Din acest motiv:

- trebuie să se cunoască modul de reprezentare a numerelor în memorie: întregi cu sau fără semn, ZCB împachetat, ASCII, cu virgulă fixă (pentru reprezentarea numerelor fracționare);
- trebuie să se facă estimarea mărimii rezultatelor şi operatorilor, pentru a se putea alege lungimea locațiilor de memorie necesară pentru stocarea lor;
- trebuie să se cunoască foarte bine modul de afectare a fanioanelor în urma unei operații, deoarece salturile condiționate se fac relativ la acestea şi, în plus, fanioanele ne pot da indicații asupra corectitudinii execuției.

4. Modul de lucru recomandat

Lucrarea de laborator a fost structurată în două părți. În prima parte sunt date două programe care nu au un scop anume, fiind alcătuite dintr-o înşiruire de instrucțiuni, grupate după anumite criterii, care ilustrează tipul de operații de interes. Primul program se ocupă de instrucțiunile aritmetice şi logice, iar al doilea, de salturile condiționate şi necondiționate.

Când se editează acest program, se vor ignora comentariile. Acestea vor trebui urmărite când se va face rularea pas cu pas a programelor, deoarece explică semnificația fiecărei instrucțiuni. De asemenea, se indică ce registre şi ce zone de memorie se modifică.

A doua parte a lucrării conține două aplicații propriu-zise care îşi propun să ilustreze rezolvarea unor probleme reale. Aici se va acorda atenție atât algoritmului cât şi sintaxei instrucțiunilor.

5. Desfăşurarea lucrării

5.1. Se lansează turbo-asamblorul TASMB şi se editează textul programului 1, fără comentarii.

5.2. Se salvează programul sursă pe disc, cu un nume oarecare, apoi se activează (**O**) opțiunea **F8 - COM FILE** şi se asamblează.

5.3. Se vizualizează lista de simboluri (**S**) şi se notează adresele de memorie ale celor patru operanzi.

5.4. Se părăseşte TASMB (**Q**) şi se lansează AFD. Se încarcă (**L**) programul salvat anterior cu extensia **COM** şi se rulează pas cu pas, urmărindu-se, în paralel, indicațiile date în comentarii.

5.6. Se repetă punctele 5.1.-5.4. pentru programele 2, 3 şi 4.

Programul 1

Observație: La toate instrucțiunile de prelucrare a datelor, trebuie să se urmărească cum sunt afectate fanioanele şi să se verifice aceasta confruntându-le cu explicațiile date în memoriul de instrucțiuni.

```
            org   100h
            mov   ax,cs
            mov   ds,ax

conv:       mov   al,opb1
            mov   bl,opb2
            xor   ah,ah       ;AH=0
            xor   bh,bh       ;BH=0
            cbw               ;converteste tipul operandului din AL,
                              ;din byte in word, prin extinderea
                              ;bitului de semn
            xchg  ax,bx
            cbw
compl:      mov   ax,bx
            neg   bx          ;complementare fata de 2
            not   ax          ;negare logica (bit cu bit)
                              ;operatiile NEG si NOT se efectueaza
                              ;asupra aceleasi valori. Diferenta
                              ;intre AX si BX ilustreaza efectul
                              ;diferit al celor doua operatii
            inc   bx          ;se urmaresc registrul BX si zona
            dec   opb2        ;de memorie de la adresa “opb2”
comp:       xor   cx,cx       ;CF=0
            cmp   opw2,24h    ;comparare prin scadere: opw2-24h
            test  opw2,20h    ;comparare prin SI logic: opw2 AND 20h
            cmp   opw2,23h    ;in general, dupa instructiunile
            cmp   opw2,20h    ;de comparare, urmeaza un salt
                              ;conditionat. De aceea trebuie urmarit
                              ;modul in care sunt afectate fanioanele
                              ;si diferenta dintre CMP si TEST

deplas:     inc   opb2
            mov   dx,opw1
            mov   cl,3
            shl   dx,1        ;deplasare logica stinga cu o pozitie
            shl   dx,cl       ;si cu 3 pozitii
            mov   dl,opb1
            shr   dl,1        ;deplasare logica dreapta cu o pozitie
            sal   dl,1        ;se remarca diferenta intre deplasarile
                              ;aritmetice si cele logice si intre
            sar   dl,1        ;deplasarile cu o pozitie si cele
            sar   dl,cl       ;cu mai multe pozitii folosind CL

rotatii:    mov   al,opb2
            inc   cl
            rol   al,1        ;se remarca diferenta dintre deplasari
                              ;si rotatii
            ror   al,1
```

```
            xor   si,si ;CF=0
            rcl   al,1
            rcr   al,1        ;se remarca diferenta dintre rotatiile
            rcr   al,cl       ;'simple' si cele prin intermediul lui
                              ;CF

op_log:     and   al,00000010b;mai intii se estimeaza
            or    al,10000000b;rezultatele celor doua operatii
                              ;si apoi se verifica rezultatul


adun: mov   bx,opw1           ;aceasta secventa simuleaza o adunare
            add   bl,byte ptr opw2
                              ;pe 16 biti din adunari pe 8 biti
            adc   bh,byte ptr opw2+1
            mov   ax,bx       ;mai intii se estimeaza rezultatul
            daa               ;in cele doua cazuri: operatori priviti
                              ;ca intregi cu semn, si fara semn.
                              ;se observa ca rezultatul obtinut
                              ;inainte de DAA, este eronat daca
                              ;operanzii sunt considerati fara semn
                              ;(apare o depasire), sau este corect
                              ;daca sunt considerati cu semn.

scad: mov   dx,opw1           ;secventa de instructiuni simuleaza
                              ;o scadere pe 16 biti,
            sub   dl,low opw2
                              ;din scaderi pe 8 biti.
            sbb   dh,high opw2
            mov   ax,dx       ;se evalueaza rezultatele in cele doua
                              ;cazuri specificate la adunare,
            das               ;apoi se verifica corectitudinea lor,
                              ;inainte de DAS

imult:      mov   ax,opw1     ;se remarca diferenta dintre
            mul   opw2        ;rezultatele obtinute in (DX)|(AX)
            mov   ax,opw1     ;pentru cele doua cazuri de inmultire.
            imul  opw2        ;pentru aceasta, rezultatul obtinut
            aam               ;dupa prima inmultire trebuie notat

impart:     mov   ax,opw1     ;se remarca diferenta dintre cele
            mov   dx,0        ;doua tipuri de impartire, observand
            div   opw2        ;diferentele intre rezultatele
            mov   ax,opw2     ;obtinute in AH si AL. Este bine
            and   dx,0        ;sa se observe necesitatea de a scrie
            idiv  opw1        ;0 in DX, inainte de impartire
            int   20h

opw1: dw    0fffdh            ;65533 valoare fara semn
                              ;-3 valoare cu semn
opw2: dw    23h               ;35 valoare cu sau fara semn
opb1: db    0feh              ;254 valoare fara semn
                              ;-2  valoare cu semn
opb2: db    23h               ;35  valoare cu sau fara semn.
```

Programul 2

Observații:

- În programele scrise în limbaj de asamblare, datele se declară în general la sfârşit. Dacă se fac declarațiile la început, procesorul va interpreta aceste date ca pe nişte instrucțiuni. Efectul acestora este greu de prevăzut. Pentru a evita acest lucru este necesar ca programul să înceapă cu un salt necondiționat la prima instrucțiune.
- În programul următor, pentru a uşura rularea pas cu pas, după fiecare salt condiționat se revine la instrucțiunea imediat următoare, cu ajutorul unui salt necondiționat indirect prin **BX**. De aceea, se poate observa în program că, înainte de salturile condiționate, se pregăteşte registrul **BX**.

```
            org   100h
            jmp   start
op1:        db    0feh        ; valoare fara semn 253
                                ; valoare cu semn -2
op2:        db      3

start:      mov   ax,cs
            mov   ds,ax
                                ; SALTURI CONDITIONATE
            mov   al,op1
            lea   bx,adr1
            cmp   al,op2        ;in functie de starea fanioanelor
                                ;dupa CMP se estimeaza, mai intii,
            ja    fin           ;efectul celor doua instructiuni
                                ;JA si JG, iar apoi se verifica
adr1: jg    fin

            mov   al,op2        ;in functie de starea fanioanelor
            lea   bx,adr2       ;dupa CMP se estimeaza, mai intii,
            cmp   al,op1        ;efectul celor doua instructiuni

            jb    fin           ; JB si JL, iar apoi se verifica
adr2: jl    fin

            lea   bx,adr3
            cmp   al,op2
            jz    fin
adr3: lea   bx,adr4
            and   al,0feh
            jnp   fin
adr4: jp    fin
            jmp   ies           ; SALT NECONDITIONAT
fin:        jmp   bx            ; SALT INDIRECT

ies:        int   20h
            end
```

Programul 3

Programul simulează o înmulțire pe 16 biți din înmulțiri pe 8 biți. Rezultatul, care poate avea maximum 32 de biți, se memorează în cuvintele de memorie '**hrez**' şi '**lrez**'.

Algoritmul de calcul este următorul:

- Un număr de 16 biți se poate reprezenta astfel:
  XYZTh = XY∗256 + ZT
  (**XY** este conținutul octetului „high” iar **ZT** este conținutul octetului „low”).
- Atunci:
  ABCDh∗XYZTh=AB∗XY∗256∗256 + (AB∗ZT+XY∗CD)∗256 + CD∗ZT.
- Operațiile de înmulțire cu puteri ale lui doi se fac prin deplasări. La sfârşitul programului se face direct înmulțirea pe 16 biți a celor doi operanzi, pentru a putea verifica corectitudinea algoritmului.

```
            org   100h
start:      mov   al,byte ptr op1+1
            mul   byte ptr [op2+1]
            mov   hrez,ax

            mov   al,byte ptr op1+1
            mul   byte ptr op2
            mov   dx,ax
            mov   cl,8
            sal   ax,cl
            mov   cl,8
            shr   dx,cl
            add   lrez,ax
            adc   hrez,0
            add   hrez,dx

            mov   al,byte ptr op2+1
            mul   byte ptr op1
            mov   dx,ax
            mov   cl,8
            shl   ax,cl
            mov   cl,8
            shr   dx,cl
            add   lrez,ax
            adc   hrez,0
            add   hrez,dx

            mov   al,byte ptr op1
            mul   byte ptr op2
            add   lrez,ax
            adc   hrez,0

            mov   ax,lrez
            mov   dx,hrez

            mov   ax,op1
            mul   op2
```

```
            int   20h
op1:        dw    300
op2:        dw    200
lrez: dw    ?
hrez: dw    ?s
            end
```

Programul 4

Se consideră un şir de maximum 16 cuvinte care ar trebui să fie identice între ele şi egale cu o valoare de referință '**ref**'. Algoritmul îşi propune să găsească toate elementele diferite şi să specifice adresa lor.

Zona de memorie de la adresa '**nrdif**' va conține numerele din şir diferite de referință, iar '**poz**' adresele acestor numere.

```
            org   100h
            mov   ax,cs
            mov   ds,ax
            lea   di,sir
            cld
            mov   bx,0ffffh
            mov   ax,ref
            mov   cx,(nrdif-sir)/2

et1:        repz  scasw
            jz    fin
            inc   bx
            mov   dx,[di-2]
            shl   bx,1
            mov   [nrdif+bx],dx
            mov   dx,di
            sub   dx,2
            mov   [poz+bx],dx
            shr   bx,1
            jmp   et1
fin:        inc   bx
            int   20h

sir         dw    1,1,2,3,1,1,1,1,4,1,5,1,1,1,4,1
nrdif       dw    16 dup (0ffffh)
poz         dw    16 dup (0ffffh)
ref         equ   1

            end
```