KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

Kompiuterių architektūros antrasis laboratorinis darbas

„Procesorių sistemos komandos architektūra“

Atliko:

IFF 6/8 grupės studentas

Tadas Laurinaitis

Priėmė

Lekt. T. Bakšys

1. ĮŽANGA

Buvo gauta užduotis suprojektuoti duotas funkcijas naudojantis „Assembly“ ir emu8088 programa, atsižvelgiant į kiekvieno kintamojo duomenų formatą, patikrinti jų veikimą bei visą veiksmų eigą aprašyti ataskaitoje.

329	Laurinaitis Tadas	IFF-6/8	2	9	Be ženklo

Pav. Nr. 1 – individualios užduoties variantai, ženklas.

2.

$$y = \begin{cases} a + c^2 & \text{, jei } c = 2x \\ |b - 2x| & \text{, jei } c < 2x \\ \left]\frac{3c+x}{c-2x}\right[& \text{, jei } c > 2x \end{cases}$$

Pav. Nr. 2 – Variantas Nr. 2 bei jo funkcijos.

Užd. nr.	A	B	C	X	Y
1	b	b	w	w	w
2	b	w	b	w	w
3	b	w	w	b	w
4	b	w	w	w	b
5	w	b	b	w	w
6	w	b	w	b	w
7	w	b	w	w	b
8	w	w	b	b	w
9	w	w	b	w	b
10	w	w	w	b	b
11	b	b	b	w	w
12	b	w	b	b	w
13	b	w	w	b	b
14	w	b	b	b	w
15	w	w	b	b	b
16	b	b	w	b	w

Pav. Nr. 3 – mėlynai apibrauktas naudotas duomenų formatas, pagal gautą individualią užduotį.

2. TEORIJA

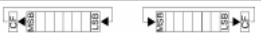
Naudotas „x86 procesoriaus komandų sąrašas“:

PERSIUNTIMO				VĖLEVĖLĖS							
Pavad.	Komentaras	Kodas	Operacija	O	D	I	T	S	Z	A	P	C
MOV	Perkelti (kopijuoti)	MOV Op1, Op2	Op1:=Op2									
XCHG	Sukeisti	XCHG Op1, Op2	Op1:=Op2 , Op2:=Op1									
STC	Nust. Carry	STC	CF:=1									1
CLC	Išval. Carry	CLC	CF:=0									0
CMC	NE Carry	CMC	CF:=¬ CF									±
STD	Nust. Kryptį	STD	DF:=1		1							
CLD	Išval. Kryptį	CLD	DF:=0		0							
STI	Nust. Pertraukimus	STI	IF:=1			1						
CLI	Atš. Pertraukimus	CLI	IF:=0			0						
PUSH	Įstumti į steką	PUSH Op	DEC SP, [SP]:=Op									
PUSHF	Įstumti į FLAGS	PUSHF	O,D,I,T,S,Z,A,P,C									
PUSHA	Įstumti visus reg.	PUSHA	AX,CX,DX,BX,SP,BP,SI,DI									
POP	Ištraukti iš steko	POP Op1	Op1:= [SP], INC SP									
POPF	Ištraukti FLAGS	PUSHF	O,D,I,T,S,Z,A,P,C	±	±	±	±	±	±	±	±	±
POPA	Ištraukti visus reg.	POPA	DI,SI,BP,SP,BX,DX,CX,AX									
CBW	Baitas ↦ žod.	CBW	AX:=AL (su ženklu)									
CWD	Žod. ↦ dvigubas	CWD	DX:AX:=AX (su ženklu)	±				±	±	±	±	±
CWDE	Žod. ↦ išpl. dvig.	CWDE	EAX:=AX (su ženklu)									
IN *i*	Įvedimas	IN Op1, Prievadas	AL,AX,EAX:=Prievadas									
OUT *i*	Išvedimas	OUT Prievadas, Op1	Prievadas:=AL,AX,EAX									

i-skaityti aprašą. Vėlevėlės: ± = pakeičiamos šia instrukcija ? = nežinomos po šios instrukcijos vykdymo

64

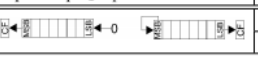
Pav. Nr. 4 – „x86 procesoriaus komandų sąrašas“ pirma dalis.

ARITMETINĖS				VĖLEVĖLĖS								
Pavad.	Komentaras	Kodas	Operacija	O	D	I	T	S	Z	A	P	C
ADD	sudėtis	ADD Op1, Op2	Op1:=Op1+Op2	±				±	±	±	±	±
ADC	sudėtis su pernaša	ADC Op1, Op2	Op1:=Op1+Op2+CF	±				±	±	±	±	±
SUB	atimtis	SUB Op1, Op2	Op1:=Op1-Op2	±				±	±	±	±	±
SBB	atimtis su pernaša	SBB Op1, Op2	Op1:=Op1-(Op2+CF)	±				±	±	±	±	±
DIV	dalyba (be ženklo)	DIV Op	Op=baitas: AL:=AX/Op AH:=liek.	?				?	?	?	?	?
DIV	dalyba (be ženklo)	DIV Op	Op=žodis: AX:=DX:AX/Op DX:=liek.	?				?	?	?	?	?
IDIV	dalyba (su ženklu)	IDIV Op	Op=baitas: AL:=AX/Op AH:=liek.	?				?	?	?	?	?
IDIV	dalyba (su ženklu)	IDIV Op	Op=žodis: AX:=DX:AX/Op DX:=liek.	?				?	?	?	?	?
MUL	daugyba (be ženklo)	MUL Op	Op=baitas: AX:=AL*Op jei AH=0 •	±				?	?	?	?	?
MUL	daugyba (be ženklo)	MUL Op	Op=žodis: DX:AX:=AX*Op jei DX=0 •	±				?	?	?	?	?
IMUL*i*	daugyba (su ženklu)	IMUL Op	Op=baitas: AX:=AL*Op •	±				?	?	?	?	?
IMUL	daugyba (su ženklu)	IMUL Op	Op=žodis: DX:AX:=AX*Op •	±				?	?	?	?	?
INC	padidinti	INC Op	Op:=Op + 1 (CF nesikeičia!)	±				±	±	±	±	
DEC	sumažinti	DEC Op	Op:=Op - 1 (CF nesikeičia!)	±				±	±	±	±	
CMP	palyginti	CMP Op1, Op2	Op1-Op2	±				±	±	±	±	±
SAL	aritm. p. į kairę	SAL Op, dydis		*i*				±	±	?	±	±
SAR	arimt. p. į dešinę	SAR Op, dydis		*i*				±	±	?	±	±
RCL	cikl. p. į kairę su C	RCL Op, dydis		*i*								±
RCR	cikl. p. į dešinę su C	RCR Op, dydis		*i*								±
ROL	cikl. p. į kairę be C	SAL Op, dydis		*i*								±
ROR	cikl. p. į dešinę be C	SAR Op, dydis		*i*								±

i-skaityti aprašą. • Tuomet CF:=0, OF:=0 kitu atveju CF:=1, OF:=1

95

Pav. Nr. 5 – „x86 procesoriaus komandų sąrašas“ antra dalis.

LOGINĖS				VĖLEVĖLĖS								
Pavad.	**Komentaras**	**Kodas**	**Operacija**	**O**	**D**	**I**	**T**	**S**	**Z**	**A**	**P**	**C**
NEG	neigimas (pap.k)	NEG Op	Op:=0-Op, jei Op=0, CF:=0	±				±	±	±	±	±
NOT	bitų inversija	NOT Op	Op:=¬Op (invertuoti bitai)									
AND	loginis IR	AND Op1, Op2	Op1:=Op1∨Op2	0				±	±	?	±	0
OR	loginis ARBA	OR Op1, Op2	Op1:=Op1∧Op2	0				±	±	?	±	0
XOR	suma moduliu 2	XOR Op1, Op2	Op1:=Op1⊕Op2	0				±	±	?	±	0
SHL	p. į kairę	SHL Op, dydis		*i*				±	±	?	±	±
SHR	p. į dešinę	SHR Op, dydis		*i*				±	±	?	±	±

ĮVAIRIOS				VĖLEVĖLĖS								
Pavad.	**Komentaras**	**Kodas**	**Operacija**	**O**	**D**	**I**	**T**	**S**	**Z**	**A**	**P**	**C**
NOP	nėra operacijos	NOP	Nėra operacijos									
LEA	užkrauti adresą	LEA Op1, Op2	Op1:= Op2 adresas									
INT	petraukimas	INT Nr	pertraukia programą			0	0					

ŠUOLIAI (vėlevėlės nesikeičia)							
Pavad.	**Komentaras**	**Kodas**	**Operacija**	**Pavad.**	**Komentaras**	**Kodas**	**Operacija**
CALL	kviesti proc.	CALL Proc		RET	Grįžti iš proc.s	RET	
JMP	besąlyginis	JMP tikslas					
JE	jei lygu	JE tikslas	(≡JZ)	JNE	jei ne lygu	JNE tikslas	(≡JNZ)
JZ	jei nulis	JZ tikslas	(≡JE)	JNZ	jei ne nulis	JNZ tikslas	(≡JNE)
JCXZ	jei CX nulis	JCXZ tikslas					
JP	jei paritetas lyg.	JP tikslas	(≡JPE)	JNP	jei paritetas nelyg.	JNP tikslas	(≡JPO)
JPE	jei paritetas lyg.	JPE tikslas	(≡JP)	JPO	jei paritetas nelyg.	JPO tikslas	(≡JNP)

96

Pav. Nr. 6 – „x86 procesoriaus komandų sąrašas“ trečia dalis.

ŠUOLIAI be ženklo				ŠUOLIAI su ženklu			
Pavad.	**Komentaras**	**Kodas**	**Ekviv.**	**Pavad.**	**Komentaras**	**Kodas**	**Ekviv.**
JA	jei daugiau	JA tikslas	(≡JNBE)	JG	jei daugiau	JG tikslas	(≡JNLE)
JAE	jei daugiau ar lygu	JAE tikslas	(≡JNB≡JNC)	JGE	jei daugiau ar lygu	JGE tikslas	(≡JNL)
JB	jei mažiau	JB tikslas	(≡JNAE≡JC)	JL	jei mažiau	JL tikslas	(≡JNGE)
JBE	jei mažiau ar lygu	JBE tikslas	(≡JNA)	JLE	jei mažiau ar lygu	JLE tikslas	(≡JNG)
JNA	jei ne daugiau	JNA tikslas	(≡JBE)	JNG	jei ne daugiau	JNG tikslas	(≡JLE)
JNAE	jei mažiau ar lygu	JNAE tikslas	(≡JB≡JC)	JNGE	jei mažiau ar lygu	JNGE tikslas	(≡JL)
JNB	jei ne mažiau	JNB tikslas	(≡JAE≡JNC)	JNL	jei ne mažiau	JNL tikslas	(≡JGE)
JNBE	jei daugiau	JNBE tikslas	(≡JA)	JNLE	jei daugiau	JNLE tikslas	(≡JG)
JC	jei pernaša	JC tikslas		JO	jei perpilda	JO tikslas	
JNC	jei nėra pernašos	JNC tikslas		JNO	jei nėra perpildos	JNO tikslas	
				JS	jei yra ženklas (-)	JS tikslas	
				JNS	jei nėra ženklo (+)	JNS tikslas	

97

Pav. Nr. 7 – „x86 procesoriaus komandų sąrašas“ ketvirta dalis.

3. REALIZACIJA

```
;
; suskaiciuoti      /   a + c^2          , kai c = 2x
;              y = |   |b-2x|           , kai c < 2x
;                  \   ](3c+x)/(c-2x)[  , kai c > 2x
; skaiciai be zenklo
; Duomenys a - w, b - w, c - b, x - w, y - b

stekas  SEGMENT STACK
DB 256 DUP(0)
stekas  ENDS

;------------Duomenys-----------------------
duom    SEGMENT
a    DW 10  ;10000 ; perpildymo situacijai
b    DW 12
c    DB 4
x    DW 1,2,3,41,50,100,6000,60000
kiek    = ($-x)/2
y    DB kiek dup(0AAh)
isvb    DB 'x=',6 dup (?), ' y=',6 dup (?), 0Dh, 0Ah, '$'
perp    DB 'Perpildymas', 0Dh, 0Ah, '$'
daln    DB 'Dalyba is nulio', 0Dh, 0Ah, '$'
netb    DB 'Netelpa i baita', 0Dh, 0Ah, '$'
;=====================================================
minus   DB 'Skaicius yra mazesnis uz 0', 0Dh, 0Ah, '$'
;=====================================================
spausk  DB 'Skaiciavimas baigtas, spausk bet kuri klavisa,', 0Dh, 0Ah, '$'
duom    ENDS

;----------Procesai-----------------------
prog    SEGMENT
assume ss:stekas, ds:duom, cs:prog

pr:
MOV ax, duom
MOV ds, ax
XOR si, si      ; (suma mod 2) si = 0
XOR di, di      ; di = 0

c_pr:
MOV cx, kiek
JCXZ pab

cikl:
MOV ax, 2         |
MOV bl, c
XOR bh, bh
MUL x[si]
CMP bx, ax
JB f2  ; c < 2x
JA f3  ; c > 2x

;==== a + c^2, kai c = 2x ====
f1:
MOV al, c
XOR ah, ah
MUL c    ; dx:ax=c^2
JC kl1  ; sandauga netilpo i ax
XCHG ax, dx
MOV ax, a
;XOR ah, ah
ADD dx, ax    ; c^2+a
JC kl1  ; suma netilpo i ax
MOV ax, dx
XOR dx, dx
;DIV bx    ; ax=rez
JMP re
```

Pav. Nr. 8 – programos kodo dalis

```
;==== b - 2x, kai c < 2x ====
f2:
MOV ax, x[si]
MOV dx, 2
MUL dx
JC kl1 ; perpildymas
XCHG ax, bx
XOR ax, ax
MOV ax, b
;====
CMP ax, bx
JB kl4 ; atsakymas yra maziau uz 0
;====
SUB ax, bx
XOR bx, bx
JMP re

;==== ](3c + x)/(c - 2x)[, kai c > 2x ====
f3:
MOV ax, 3
MUL c           ; 3 * c
JC kl1 ;ar ne perpildytas
MOV dx, x[si]
ADD ax, dx      ;3c + x
JC kl1 ;ar ne perpildytas
XOR bx, bx
XCHG  ax, bx
MOV ax, 2
MUL dx        ; 2 * x  <------- automatiskai nuresetina dx kazkodel :/
XOR dx, dx
XCHG ax, dx
MOV al, c
XOR ah, ah
SUB ax, dx    ;c - 2x
CMP ax, 0
JZ kl2
XOR dx, dx
XCHG ax, bx
DIV bx
JMP re

re:
CMP ah, 0     ;ar telpa rezultatasi baita
JE ger
JMP kl3

ger:    MOV y[di], al
INC si
INC si
INC di
LOOP cikl

pab:
;rezultatu isvedimas i ekrana
;============================
XOR si, si
XOR di, di
MOV cx, kiek
JCXZ is_pab

is_cikl:
MOV ax, x[si]  ; isvedamas skaicius x yra ax reg.
PUSH ax
MOV bx, offset isvb+2
PUSH bx
CALL binasc
MOV al, y[di]
XOR ah, ah        ; isvedamas skaicius y yra ax reg.
PUSH ax
MOV bx, offset isvb+11
PUSH bx
CALL binasc
```

Pav. Nr. 9 – programos kodo dalis

```
is_pab:
;===== PAUZE ====================
;===== paspausti bet kuri klavisa ===
LEA dx, spausk
MOV ah, 9
INT 21h
MOV ah, 0
INT 16h
;============================
MOV ah, 4Ch    ; programos pabaiga, grizti i OS
INT 21h
;============================

kl1:    LEA dx, perp
MOV ah, 9
INT 21h
XOR al, al
JMP ger

kl2:    LEA dx, daln
MOV ah, 9
INT 21h
XOR al, al
JMP ger

kl3:    LEA dx, netb
MOV ah, 9
INT 21h
XOR al, al
JMP ger
;==== MAZIAU UZ 0 ====
kl4:    LEA dx, minus
MOV ah, 9
INT 21h
XOR al, al
JMP ger
;======================
; skaiciu vercia i desimtaine sist. ir issaugo
; ASCII kode. Parametrai perduodami per steka
; Pirmasis parametras ([bp+6])- verciamas skaicius
; Antrasis parametras ([bp+4])- vieta rezultatui

binasc    PROC NEAR
PUSH bp
MOV bp, sp
; naudojamu registru issaugojimas
PUSHA
; rezultato eilute uzpildome tarpais
MOV cx, 6
MOV bx, [bp+4]
tarp:   MOV byte ptr[bx], ' '
INC bx
LOOP tarp
; skaicius paruosiamas dalybai is 10
MOV ax, [bp+6]
MOV si, 10

val:    XOR dx, dx
DIV si
; gauta liekana verciame i ASCII koda
ADD dx, '0'    ; galima--> ADD dx, 30h
; irasome skaitmeni i eilutes pabaiga
DEC bx
MOV [bx], dl
; skaiciuojame pervestu simboliu kieki
INC cx
; ar dar reikia kartoti dalyba?
CMP ax, 0
JNZ val

POPA
POP bp
RET
binasc    ENDP
prog    ENDS
END pr
```

Pav. Nr. 10 – programos kodo dalis

Sprendimo eiga: Pirmas žingsnis buvo peržiūrėti pavyzdinę užduotį ir suvokti pačius pradmenis. Po to atsižvelgiant į kintamųjų formatus, apsirašiau kintamuosius, bei paeiliui koregavau duotą pavyzdinę užduotį, pradedant nuo ciklo funkcijos, kurioje atsižvelgiama į lygties sąlygą ir peršokama į reikiamą funkciją, baigiant trečios lygties skaičiavimo funkcija. Sprendimo metu susidūrus su klaida, suemuliuodavau kodą ir eidavau pažingsniui stengdamasis išsiaiškinti padarytą klaidą.

4. REZULTATAI

```
;
; suskaiciuoti      /   a + c^2          , kai c = 2x
;              y = |   |b-2x|           , kai c < 2x
;                  \   ](3c+x)/(c-2x)[  , kai c > 2x
; skaiciai be zenklo
; Duomenys a - w, b - w, c - b, x - w, y - b

stekas  SEGMENT STACK
DB 256 DUP(0)
stekas  ENDS

;------------Duomenys-----------------------
duom    SEGMENT
a     DW 10   ;10000 ; perpildymo situacijai
b     DW 150
c     DB 10
x     DW 1,2,3,8,7,13,19
```

Pav. Nr. 11 – sprendžiamos lygtys, jų sąlygos, kintamųjų formatai bei reikšmės.

```
x=      1 y=      3
x=      2 y=      5
x=      3 y=      8
x=      8 y=    134
x=      7 y=    136
x=     13 y=    124
x=     19 y=    112
Skaiciavimas baigtas, spausk bet kuri klavisa,
```

Pav. Nr. 12 – suemuliavus ir paleidus programą gauti rezultatai.

Suskaičiavus pačiam ir patikrinus rezultatus su gautais rezultatais galima teigti, jog programa skaičiuoja teisingai.

5. IŠVADOS

Šio laboratorinio darbo metu sėkmingai susipažinau su Assembly pradmenimis, taip pat susipažinau su emu8088 aplinka bei valdymu. Individualioje užduotyje nurodytos užduotys buvo sėkmingai išspręstos, išnagrinėtos bei patikrintas jų teisingumas.