Sistemski softver

Domaći zadatak 2019.

Uroš Zekić 2015/0185

Opis zadatka:

Zadatak je napraviti dvoprolazni asembler koji će biti upotrebljen na 16-bitnom dvoadresnom procesoru sa Von-Neuman arhitekturom čija adresibilna jedinica je jedan bajt, a raspored bajtova u reči je little-endian. Procesor poseduje osam opštenamenskih 16-bitnih registara(r0-7). Moguće je pristupati višem ili nižem bajtu svakog od ovih registara. Registar r7 se koristi kao pc registar,a registar r6 se koristi kao sp registar. Takođe postoji i psw registar. Treba napraviti podršku za asemblerske direktive .equ, .word, .byte, .skip, .align, .global i .extern koje imaju identične funkcionalnosti kao u okviru GNU asemblera. Prilikom generisanja izlaza potrebno je voditi se principima GNU asemblera.

Rešenje:

Preko komandne linije asembleru se prosleđuju parametri, ulazni fajl sa kodom koji treba obraditi, i izlazni fajl koji treba generisati. Metodi FirstRun se prosleđuje ulazni fajl, kako bi se izvršio prvi prolaz asemblera, a metodi SecondRun i ulazni i izlazni,kako bi se izvršio drugi prolaz i generisao izlazni fajl.

U prvom prolazu se popunjava tabela simbola,uz obradu neizračunljivih simbola nakon prvog prolaza,po algoritmu sa vežbi.Moguće je equ direktivom zadati vrednost simbolu koju treba sračunati(jednostavni izraz, operacija sabiranja ili oduzimanja,dva operanda). Takođe se vrši provera ulaznog fajla, da li ima greški u kodu, ili nedozvoljenih instrukcija ili direktiva.

U drugom prolazu se vrše provere da li je neki simbol ostao nedefinisan, i taj slučaj obrađuje kao greška. Kreiraju se relokacioni zapisi tamo gde je potrebno, i generiše izlazni fajl.

Prevođenje:

Komande neophodne,kako bi se virtuelna mašina koja se koristi (ubuntu 12.04 lts) prilagodila za rad sa ovim asemblerom su:

1. sudo apt-get update

2. sudo apt-get install build-essential software-properties-common -y

3. sudo add-apt-repository ppa:ubuntu-toolchain-r/test -y

4. sudo apt-get update

5. sudo apt-get install gcc-snapshot -y

6. sudo apt-get update

7. sudo apt-get install gcc-6 g++-6 -y

8. sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-6 60 --slave /usr/bin/g++ g++ /usr/bin/g++-6

9. sudo apt-get install gcc-4.8 g++-4.8 -y

10. sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.8 60 --slave /usr/bin/g++ g++ /usr/bin/g++-4.8;

11. sudo add-apt-repository ppa:ubuntu-toolchain-r/test  
12. sudo apt-get update  
13. sudo apt-get install gcc-4.9  
14. sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.9 50  
sudo apt-get install g++-4.9  
15. sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.9 50

Potrebno je napraviti folder u virtuelnoj mašini u kojem će biti smešten projekat, a zatim u terminalu podesiti trenutni direktorijum na taj folder. (cd folder)

Cpp fajlove je neophodno prevesti u s fajlove.(primer komande)

g++-6 -std=c++0x -o assembler.s assembler.cpp -S

Od dobijenih s fajlova, treba napraviti izvršni program komandom:

g++-6 -o projekat Instruction.s RelocationTableEntry.s SymbolTableEntry.s TNSEntry.s assembler.s main.s

Prilikom pokretanja projekta treba mu proslediti argumenta, komandom:

./projekat input.txt output.txt

Testovi:

1.test

.data

.word A

.byte 0x5,3

.extern f

.global G

A: .skip 2

.text

mov r1,3

push r1

addb r2l,0x2

.equ b,A-1

sub r3,f

loop:

add r1,1

cmp r1,5

jeq $loop

.align 4

push G

.equ U,G-1

.byte G-U

.bss

.skip 2

G: .skip 8

.section sekc2

or r1 r2

iret

.end

1.izlaz

.data

0x04 0x00

0x05 0x03

0x00 0x00

.text

0x24 0x22 0x00 0x03 0x00

0x8c 0x22

0x2c 0x24 0x00 0x02

0x34 0x26 0xa0 0x00 0x00

0x2c 0x22 0x00 0x01 0x00

0x4c 0x22 0x00 0x05 0x00

0xa4 0x8e 0xf2 0xff

0x00 0x00

0x8c 0xa0 0x00 0x00

0x01

.bss

0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

sekc2

0x64 0x22 0x24

0xc8

SymbolTable

name section value visibility num

und und 0x0 l 0

.data .data 0x0 l 1

f und 0x0 g 2

A .data 0x4 l 3

.text .text 0x0 l 4

b abs 0x3 l 5

loop .text 0x10 l 6

.bss .bss 0x0 l 7

G .bss 0x2 g 8

sekc2 sekc2 0x0 l 9

U abs 0x1 l 10

#.ret..data

offset type num

0x0 R\_386\_16 1

#.ret..text

offset type num

0xe R\_386\_16 2

0x22 R\_386\_16 8

0x24 R\_386\_16 8

#.ret..bss

offset type num

#.ret.sekc2

offset type num

2.test

.data

.extern C

.global gl,zl

bytes: .byte 13, 0xa0, 22,0x04

.skip 4

.text

.equ j, bytes + 5

mov r1[3], r2[0]

prog: mov r3, \*100

add r4,r3

mov r3, \*0xF0F1

add r3, C

ret

.section nova rwx

push j

.equ k,labela

.equ z,k+1

push r1

labela:

jmp $bytes

push sp

push psw

call $prog

zl: mov r1,0x1F

sub r2,zl

.bss

.skip 4

gl: .skip 6

.end

2.izlaz

.data

0x0d 0xa0 0x16 0x04

0x00 0x00 0x00 0x00

.text

0x24 0x62 0x03 0x44

0x24 0x26 0xa0 0x64 0x00

0x2c 0x28 0x26

0x24 0x26 0xa0 0xF1 0xF0

0x2c 0x26 0xa0 0x00 0x00

0xc0

nova

0x8c 0xa0 0x05 0x00

0x8c 0x22

0x9c 0x8e 0xfe 0xff

0x8c 0x2c

0x8c 0x3e

0xbc 0x8e 0x02 0x00

0x24 0x22 0x00 0x1F 0x00

0x34 0x24 0xa0 0x00 0x00

.bss

0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00 0x00 0x00

SymbolTable

name section value visibility num

und und 0x0 l 0

.data .data 0x0 l 1

C und 0x0 g 2

bytes .data 0x0 l 3

.text .text 0x0 l 4

j abs 0x5 l 5

prog .text 0x4 l 6

nova nova 0x0 l 7

labela nova 0x6 l 8

zl nova 0x12 g 9

.bss .bss 0x0 l 10

gl .bss 0x4 g 11

k abs 0x6 l 12

z abs 0x7 l 13

#.ret..data

offset type num

#.ret..text

offset type num

0x14 R\_386\_16 2

#.ret.nova

offset type num

0x8 R\_386\_PC16 1

0x10 R\_386\_PC16 4

0x1a R\_386\_16 9

#.ret..bss

offset type num

3.test

.data

.word a,b+1

b: .byte 8,0x02,0x05,9,12

.skip 4

c: .equ d, a

.section sekcija x

int 20

and r1,r3

shl r1,2

shr r3,0x4

pop r3

push sp

mov sp,0x100

mov sp[0],c

mov sp[2],15

pop sp

xor r3,d

mul r1,d

jmp $prog

.text

pop psw

prog: mov r1,&b

add r1,4

push r1

pop r2

sub r2,0x0b

a: push r1

.section writable

.skip 4

.align 8

.byte 15,0xff,d+3

.end

.data

0x15 0x00

0x05 0x00

0x08 0x02 0x05 0x09 0x0c

0x00 0x00 0x00 0x00

sekcija

0x1c 0x00 0x14 0x00

0x5c 0x22 0x26

0x7c 0x22 0x00 0x02 0x00

0x84 0x26 0x00 0x04 0x00

0x94 0x26

0x8c 0x2c

0x24 0x2c 0x00 0x00 0x01

0x24 0x4c 0xa0 0x0d 0x00

0x24 0x6c 0x02 0x00 0x0f 0x00

0x94 0x2c

0x3c 0x22 0xa0 0x15 0x00

0x9c 0x8e 0x00 0x00

.text

0x94 0x3e

0x24 0x22 0x00 0x04 0x00

0x2c 0x22 0x00 0x04 0x00

0x8c 0x22

0x94 0x24

0x34 0x24 0x00 0x0b 0x00

0x8c 0x22

writable

0x00 0x00 0x00 0x00

0x00 0x00 0x00 0x00

0x0f 0xff 0x18

SymbolTable

name section value visibility num

und und 0x0 l 0

.data .data 0x0 l 1

b .data 0x4 l 2

c .data 0xd l 3

sekcija sekcija 0x0 l 4

.text .text 0x0 l 5

prog .text 0x2 l 6

a .text 0x15 l 7

writable writable 0x0 l 8

d abs 0x15 l 9

#.ret..data

offset type num

0x0 R\_386\_16 5

0x2 R\_386\_16 1

#.ret.sekcija

offset type num

0x1d R\_386\_16 1

0x2e R\_386\_PC16 5

#.ret..text

offset type num

0x5 R\_386\_16 1

#.ret.writable

offset type num

4.test

.data

.align 8

.byte 2,3,0x1

w: .skip 2

.equ k, w + 3

.section novasekcija rw

.byte k+w

.skip 10

c: .word c

niz: .byte 4,c,w

.text

push c

a: jmp c

int 10

add r3,\*150

div r4,2

mul r4,niz

jgt $a

cmp r3,3

jeq a

jmp $z

.section druga rwx

.skip 10

z: push sp

push pc

call a

.bss

.end

4.izlaz

.data

0x02 0x03 0x01

0x00 0x00

novasekcija

0x09

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x0b 0x00

0x04 0x0b

0x03

.text

0x8c 0xa0 0x0b 0x00

0x9c 0xa0 0x0b 0x00

0x1c 0x00 0x0a 0x00

0x2c 0x26 0xa0 0x96 0x00

0x44 0x28 0x00 0x02 0x00

0x3c 0x28 0xa0 0x0d 0x00

0xb4 0x8e 0xe5 0xff

0x4c 0x26 0x00 0x03 0x00

0xa4 0xa0 0x04 0x00

0x9c 0x8e 0x08 0x00

druga

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x8c 0x2c

0x8c 0x2e

0xbc 0xa0 0x04 0x00

.bss

SymbolTable

name section value visibility num

und und 0x0 l 0

.data .data 0x0 l 1

w .data 0x3 l 2

k abs 0x6 l 3

novasekcija novasekcija 0x0 l 4

c novasekcija 0xb l 5

niz novasekcija 0xd l 6

.text .text 0x0 l 7

a .text 0x4 l 8

druga druga 0x0 l 9

z druga 0xa l 10

.bss .bss 0x0 l 11

#.ret..data

offset type num

#.ret.novasekcija

offset type num

0x0 R\_386\_16 1

0xb R\_386\_16 4

0xe R\_386\_16 4

0xf R\_386\_16 1

#.ret..text

offset type num

0x2 R\_386\_16 4

0x6 R\_386\_16 4

0x19 R\_386\_16 4

0x26 R\_386\_16 7

0x2a R\_386\_PC16 9

#.ret.druga

offset type num

0x10 R\_386\_16 7

#.ret..bss

offset type num