\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Електротехнички факултет у Београду

Катедра за рачунарску технику и информатику

***Предмет****:* Системски софтвер (13Е113СС)

***Наставници***: др Драган Бојић, др Саша Стојановић

***Асистенти:*** Марко Мићовић, дипл. инж.

***Испитни рок:*** јун 2020.

***Датум:*** 6.6.2020.

***Кандидат:*** Марко Кујунџић

***Број индекса:*** 0418/2016

***Тема:*** Интерпретативни емулатор

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Опис решења

Повезивач и Пунилац, након обраде машинског кода уграђеног у секције и пуњења програма у меморију, остављају емулатору све што му је потребно за извршавање асемблерских инструкција. Емулатор, после покретања програма, скаче у прекидну рутину у нултом улазу IV табеле, ради иницијализацију процесора постављајући све вредности у систему на иницијалне вредности, а затим у регистар pc уписује вредност \_start симбола, који уједно представља и улазну тачку у програм, одакле креће извршавање.

# Структура решења

Излаз асемблера представља објектни фајл, који уједно представља и улаз емулатора.

На почетку процеса повезивања, потребно је из објектног фајла дохватити све релевантне податке, као што су сама табела симбола, затим машински код сваке секције, као и релокациони записи за сваку секцију. Потом се прави табела симбола свих обједињених објектних фајлова, а потом долази до прављења агрегираних секција из сваког улазног објектног фајла. Распоред агрегираних секција унутар нове, обједињене секције, је произвољан.

Повезивач након тога покупи све улазне фајлове и уради њихово повезивање и прослеђује их Пуниоцу, који након тога све агрегиране секције са машинским кодом спаја у један извршиви фајл. Током тог поступка, разрешава и све релокације које су иницијално настале, уграђивањем одговарајућих вредности у машински код у зависности од типа релокације. Извршни фајл се након тога учитава у меморијски адресни простор, након чега емулатор креће са декодовањем и извршавањем инструкција, и обрадом прекида које генеришу периферије(тајмер и терминал).

Приликом процеса повезивања и пуњења долази до стандардних провера, и пријављивања грешке у случају неке недозвољене ситуације.

Неке од тих ситуација су преклапање секција у меморији, недостатак секције iv\_table која је потребан и довољан услов да би емулаторски програм уопште могао да се изврши. Такође, проверава се и присутност симбола \_start, као и то да ли су сви симболи дефинисани, као и то да ли постоји неки симбол који је остао недефинисан, што ће изазвати грешку.

Када емулатор крене са извршавањем асемблерског програма, прво долази до иницијализације целог система, свих регистара, улаза у IV табелу итд. Потом се скаче у нулти улаз IV табеле, где се у регистар РС уписује вредност симбола \_start, који представља улазну тачку (енгл. entry point) у програм који се извршава. Потом се наставља са секвенцијалном обрадом инструкција.

Периферије које се користе у раду емулатора, timer и terminal, реализоване су као две засебне нити, које се конкурентно извршавају са главном нити програма, и потом генеришу прекиде када заврше са својом рутином. Timer нит се успављује на онолико времена колико је специфицирано вредношћу timer\_cfg меморијски мапираног регистра, док terminal нит чека на унос карактера, који потом упише на адресу mem16[data\_in]. Карактери уписани на тај начин се приликом обраде прекида уписују у меморију на локацију mem16[data\_out].

Зарад бољег одзива система, коршћено је посебно заглавље Linux оперативног система <termios.h>, која омогућава да се угаси баферовање у терминалу приликом уноса карактера.

Структура адресног простора:

0xFFFF -----------------------------------------------------------

| MEMORY MAPPED REGISTERS |

0xFF00 |----------------------------------------------------------|

| |

| |

| |

| |

| |

| |

| |

| |

0x0810 | |

0x080F |----------------------------------------------------------|

| |

| STACK |

0x0010 | |

0x000F |----------------------------------------------------------|

| IVT |

0x0000 -----------------------------------------------------------|

Приказ структуре адресног простора коришћеног програма.

Покретање програма на Linux x86 архитектури:

Програм се покреће на Linux оперативном систему, прослеђивањем одговарајућих параметара у зависности од тога да ли се покреће емулатор или асемблер.

Асемблер се покреће на следећи начин:

* assembler -o output\_file.o input\_file.s

Емулатор се покреће на следећи начин:

* emulator -place<section\_name>@<start\_address> input\_file\_1.o ... input\_file\_n.o

При покретању емулатора, може се, али и не мора специфицирати стартна адреса секција у меморији. Подразумевано, IV табела се учитава у меморију на адресу 0x0000. После ње, налази се системски стек, а после њега се налази простор за учитавање корисничког програма, све до адресе 0xFF00, на којој се налазе меморијски мапирани регистри.

Извршни фајл се прави коришћењем команди make clean и make, а након тога се може покренути или на горе поменути начин, или покретањем скрипте која аутоматски покреће све тестове написане за сам програм. Покретање се врши на следећи начин, из главног директоријума:

* ./script.sh

Због разлике у формату текстуалног фајла на оперативном систему Linux и оперативном систему Windows, у програму Notepad++ потребно је урадити следеће са улазним фајловима оба програма:

* Edit => EOL Conversion => Unix(LF)

Покретање и тестови - Емулатор

Коректан рад емулатора је тестиран помоћу четири теста. Први тест захтева унос броја преко терминала, након чега долази до одбројавања до нуле. Други тест захтева унос enter карактера, наког чега се реч timer исписује на сваку временску периоду специфицирану у поставци задатка, наког чега се, после паузе од 60 секунди, реч timer исписује на сваких пола секунде. Улазни тестови, као и излази тих тестова, су дати у прилогу. Трећи тест рачуна факторијел броја који се уноси преко терминала. Као и у другом тесту, извршавање програма се започиње уносом enter карактера. Четврти тест само исписује унесене карактере са тастатуре.

Тест 1:

.global \_start

.section .text:

\_start:

jmp \_start

.section iv\_table:

.word ivt\_0

.word ivt\_1

.word ivt\_2

.word ivt\_3

.word ivt\_4

.word ivt\_5

.word ivt\_6

.word ivt\_7

.equ data\_in, 0xFF02

.equ data\_out, 0xFF00

.equ timer\_cfg, 0xFF10

.equ new\_line, 10 # ascii value for newline character

.section entries:

ivt\_0:

mov $\_start, %pc

ivt\_1:

halt

ivt\_2:

cmp $0, %r4

jeq exit\_1

cmp $0, %r0

jeq timer

cmp $0, %r0

jne inc

timer:

mov %r3, timer\_cfg

mov $1, %r0

jmp label\_t

inc:

add $1, timer\_cfg

label\_t:

mov $116, data\_out

mov $105, data\_out

mov $109, data\_out

mov $101, data\_out

mov $114, data\_out

mov $32, data\_out

iret

ivt\_3:

cmp $new\_line, data\_in # check if enter key is pressed

jeq start

jmp exit\_2

start:

mov $1, %r4

exit\_1:

iret

exit\_2:

mov $0, data\_in

iret

ivt\_4: iret

ivt\_5: iret

ivt\_6: iret

ivt\_7: iret

.end

Излаз 1:

.section entries

64 00 01 00 80 10 FF 64 00 2E 14 2E 00 8C 00 00 00 28 34 00 7C 14 8C 00 00 00 2A 34 00 69 14 64 2A 20 24

00 00 14 64 00 20 00 80 00 FF 74 00 01 00 2A 2C 6E 13 00 64 00 00 00 28 64 00 30 00 80 00 FF 64 00 0A 00

80 00 FF 08 8C 00 0A 00 80 02 FF 34 00 9E 14 7C 00 0A 00 2A 64 80 02 FF 26 74 00 30 00 26 6C 26 2A 2C 00

A3 14 64 00 01 00 28 64 00 00 00 80 02 FF 08 08 08 08 08

.section .text

4C 26 4C 2A 64 20 26 34 00 29 14 64 26 2A CC 00 0A 00 2A 84 00 0A 00 26 64 26 20 24 00 00 14 6C 00 30 00

2A 64 2A 80 00 FF 54 2A 54 26 10 2C 00 2E 14 00

.section iv\_table

33 14 3F 14 40 14 7D 14 AB 14 AC 14 AD 14 AE 14

12

12 11 10 9 8 7 6 5 4 3 2 1 0

Тест 2:

.global \_start

.section .text:

\_start:

jmp \_start

.section iv\_table:

.word ivt\_0

.word ivt\_1

.word ivt\_2

.word ivt\_3

.word ivt\_4

.word ivt\_5

.word ivt\_6

.word ivt\_7

.equ data\_in, 0xFF02

.equ data\_out, 0xFF00

.equ timer\_cfg, 0xFF10

.equ new\_line, 10 # ascii value for newline character

.section entries:

ivt\_0:

mov $\_start, %pc

ivt\_1:

halt

ivt\_2:

cmp $0, %r4

jeq exit\_1

cmp $0, %r0

jeq timer

cmp $0, %r0

jne inc

timer:

mov %r3, timer\_cfg

mov $1, %r0

jmp label\_t

inc:

add $1, timer\_cfg

label\_t:

mov $116, data\_out

mov $105, data\_out

mov $109, data\_out

mov $101, data\_out

mov $114, data\_out

mov $32, data\_out

iret

ivt\_3:

cmp $new\_line, data\_in # check if enter key is pressed

jeq start

jmp exit\_2

start:

mov $1, %r4

exit\_1:

iret

exit\_2:

mov $0, data\_in

iret

ivt\_4: iret

ivt\_5: iret

ivt\_6: iret

ivt\_7: iret

.end

Излаз 2:

.section entries

64 00 00 14 2E 00 8C 00 00 00 28 34 00 79 14 8C 00 00 00 20 34 00 25 14 8C 00 00 00 20 3C 00 33 14 64 26

80 10 FF 64 00 01 00 20 2C 00 3A 14 6C 00 01 00 80 10 FF 64 00 74 00 80 00 FF 64 00 69 00 80 00 FF 64 00

6D 00 80 00 FF 64 00 65 00 80 00 FF 64 00 72 00 80 00 FF 64 00 20 00 80 00 FF 08 8C 00 0A 00 80 02 FF 34

00 74 14 2C 00 7A 14 64 00 01 00 28 08 64 00 00 00 80 02 FF 08 08 08 08 08

.section .text

2C 00 00 14

.section iv\_table

04 14 09 14 0A 14 65 14 82 14 83 14 84 13 85 13

timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer timer

Тест 3:

.global \_start

.section .text:

print:

push %r3

push %r5

mov %r0, %r3

jeq return

mov %r3, %r5

mod $10, %r5

div $10, %r3

mov %r3, %r0

call print

add $zero, %r5

mov %r5, data\_out

return:

pop %r5

pop %r3

ret

\_start:

jmp \_start

halt

.section iv\_table:

.word ivt\_0

.word ivt\_1

.word ivt\_2

.word ivt\_3

.word ivt\_4

.word ivt\_5

.word ivt\_6

.word ivt\_7

.equ data\_in, 0xFF02

.equ data\_out, 0xFF00

.equ timer\_cfg, 0xFF10

.equ zero, 48 # ascii value for zero character

.equ space, 32 # ascii value for space character

.equ new\_line, 10 # ascii value for newline character

.section entries:

ivt\_0:

mov $1, timer\_cfg

mov $\_start, %pc

ivt\_1:

halt

ivt\_2:

iret

ivt\_3:

cmp $new\_line, data\_in # check if enter key is pressed

jeq start

mul $10, %r5

mov data\_in, %r3

sub $zero, %r3

add %r3, %r5

jmp exit\_2

start:

mov %r5, %r4

loop:

cmp $0, %r5

jeq calc

sub $0x1, %r5

cmp $0x0, %r5

jeq calc

mul %r5, %r4

jmp loop

calc:

mov %r4, %r0

cmp $0, %r4

jeq exit\_3

call print

help:

mov $0, %r5

mov $new\_line, data\_out

exit\_2:

mov $0, data\_in

iret

exit\_3:

mov $zero, data\_out

jmp help

ivt\_4: iret

ivt\_5: iret

ivt\_6: iret

ivt\_7: iret

.end

Излаз 3:

.section entries

64 00 01 00 80 10 FF 64 00 2E 14 2E 00 08 8C 00 0A 00 80 02 FF 34 00 62 14 7C 00 0A 00 2A 64 80 02 FF 26

74 00 30 00 26 6C 26 2A 2C 00 9F 14 64 2A 28 8C 00 00 00 2A 34 00 83 14 74 00 01 00 2A 8C 00 00 00 2A 34

00 83 14 7C 2A 28 2C 00 65 14 64 28 20 8C 00 00 00 28 34 00 A7 14 24 00 00 14 64 00 00 00 2A 64 00 0A 00

80 00 FF 64 00 00 00 80 02 FF 08 64 00 30 00 80 00 FF 2C 00 93 14 08 08 08 08

.section .text

4C 26 4C 2A 64 20 26 34 00 29 14 64 26 2A CC 00 0A 00 2A 84 00 0A 00 26 64 26 20 24 00 00 14 6C 00 30 00

2A 64 2A 80 00 FF 54 2A 54 26 10 2C 00 2E 14 00

.section iv\_table

33 14 3F 14 40 14 41 14 B2 14 B3 13 B4 13 B5 13

5

120

4

24

3

6

2

2

1

1

0

0

Тест 4:

.global \_start

.section .text:

\_start:

jmp \_start

.section iv\_table:

.word ivt\_0

.word ivt\_1

.word ivt\_2

.word ivt\_3

.word ivt\_4

.word ivt\_5

.word ivt\_6

.word ivt\_7

.equ data\_in, 0xFF02

.equ data\_out, 0xFF00

.section entries:

ivt\_0:

mov $\_start, %pc

ivt\_1:

halt

ivt\_2:

iret

ivt\_3:

mov data\_in, data\_out

mov $0, data\_in

iret

ivt\_4: iret

ivt\_5: iret

ivt\_6: iret

ivt\_7: iret

.end

Излаз 4:

.section entries

64 00 00 14 2E 00 08 64 80 02 FF 80 00 FF 64 00 00 00 80 02 FF 08 08 08 08 08

.section .text

2C 00 00 14

.section iv\_table

04 14 09 14 0A 14 0B 14 1A 14 1B 14 1C 14 1D 14

33 55 66 11