***~ И З В Е Ш Т А Ј ~***

*~ ПРОЈЕКАТ ИЗ СИСТЕМСКОГ СОФТВЕРА ~*

**Студент:**

*Јован Стевановић -* 0626/2015

**Београд**, 2018. године

***Задатак 1***

Ц

иљ је направити двопролазни асемблер. Први део, првог задатка се састоји од парсирања улазног текста. Унтар самог парсирања се посебно извајају делови који парсирају асемблерске директиве, асемблерске директиве и операције, операције, затим линија може бити празна или садржати неки одређен број табулација.

Парсер као класа која врши парсирање улазног текста као резултат враћа одговарајућу структуру која је попуњена информацијама дестиловане из улазног текста.

Поред класе која се бави парсирањем постоји и класа Асемблер која је централна у обављању асемблирања.

Први пролаз се обавља користећи парсер и структуре које као повратне вредности враћа парсер.

У току првог пролаза се формира табела симбола. Уколико има неких синтаксних проблема и то се пријављује. У другом пролазу се на основу табеле симбола формира објектни фајл. Ако се примети неки проблеми и то се пријављује. Формирају се релокациони записи.

Резултат асемблирања је објектни фајл који садржи табелу симбола, релокационе записе и бинарни запис превердених секција. Поред поменутих класа Парсер и Асемблер користе се још и класе Релокациона Табла са методама које баратају једном табелу.

Врше разне помоћне операције, претрагу, уметање и остале операције које би биле од користи над једном табелом. Класа Релокациона Табела садржи сруктуру која описује један улаз у ову табелу.

Поред ове класе за операције на табелама се користи још класа Табела Симбол са методама које баратају једном табелу. Врше разне помоћне операције, претрагу, уметање и остале операције које би биле од користи над једном табелом.

***Задатак 2***

Ц

иљ је направити интерпретирајући емулатор који ће емулирати процесор за асемблер из првог задатка. Интерпретер се ослања на употребу класе која обавља обраду над бинарним садржајем.

Попуњава одговарајуће структуре подацима. Срце емулатора је свич који одлучује која је операција је у питању. Затим унутар сваке од селектованих операција постоји још поједан свич који врши селектовање у зависности од типа адресирања.

Након што се изврши операција који резултује уписом у неки регистар или уписом у меморију.

После обављене операције извршава се ажирирање флегова ПСВ регистра. Након ажирирања врши се инкрементирање тајмера. После чека се одучује да ли треба се скочи на периодичан прекид или не.

Такође у прекидне рутине се може ући и ако је корисник притисну неки тастер или ако је нека инстркција некоректна. Нпр уколико је корисник покушао да дели са нулом. Корисник путем асемблерског кода може вршити и испис и то тако што ће уписати аски вредност карактера у излазни регистар.

***Тестови***

***Тест 1:*** *Попуњава се низ са вредностима од 1 до 5. Затим се тај низ помоћу рекурзивне функције обрада исписује у обрнутом редоследу.*

*Објектни фајл 1:*

***.data***

*array:*

*.skip 10*

***.text***

*.extern obrada*

*.global \_start*

*\_start:*

*moval r0, 0*

*moval r1, 1*

*moval r2, 5*

*loop\_begin:*

*cmpal r2, 0*

*jmpeq &loop\_end*

*moval r0[array], r1*

*addal r0, 2*

*addal r1, 1*

*subal r2, 1*

*jmpal &loop\_begin*

*loop\_end:*

*moval r0, &array*

*moval r1, 5*

*callal &obrada*

*jmpal 0*

***.end***

*Обектни фајл 2:*

***.text***

*.global obrada*

*obrada:*

*pushal r0*

*pushal r1*

*cmpal r1, 0*

*jmpeq &end*

*pushal r0*

*pushal r1*

*addal r0, 2*

*subal r1, 1*

*callal &obrada*

*popal r1*

*popal r0*

*moval r0, r0[0]*

*moval \*65534, r0*

*end: popal r1*

*popal r0*

*retal*

***.end***

***Тест 2:*** *Попуњава се низ елементима од 1 до 20. Затим помоћу функције обрада сумира низ и сумирани резултат се исписује на излазу.*

*Објектни фајл 1:*

***.data***

*array:*

*.skip 40*

***.text***

*.extern obrada*

*.global \_start*

*\_start:*

*moval r0, 0*

*moval r1, 1*

*moval r2, 20*

*loop\_begin:*

*cmpal r2, 0*

*jmpeq &loop\_end*

*moval r0[array], r1*

*addal r0, 2*

*addal r1, 2*

*subal r2, 1*

*jmpal &loop\_begin*

*loop\_end:*

*moval r0, &array*

*moval r1, 20*

*callal &obrada*

*jmpal 0*

***.end***

*Објектни фајл 2:*

***.data***

*x: .word 0*

***.text***

*.global obrada*

*obrada:*

*loop\_begin:*

*cmpal r1, 0*

*jmpeq &loop\_end*

*moval r2, r0[0]*

*addal x, r2*

*addal r0, 2*

*subal r1, 1*

*jmpal &loop\_begin*

*loop\_end:*

*moval r0, x*

*moval \*65534, r0*

*retal*

***.end***

***Тест 3:*** *Попуњава се низ вредностима од 1 до 39 са кораком 2. Затим се позива метода обрада\_замени која обрће елементе низа, а затим се позива метода обрада\_испис која исписује елементе сваки у једном реду.*

*Објектни фајл 1:*

***.data***

*array:*

*.skip 40*

***.text***

*.extern obrada\_zamena*

*.extern obrada\_ispis*

*.global \_start*

*\_start:*

*moval r0, 0*

*moval r1, 1*

*moval r2, 20*

*loop\_begin:*

*cmpal r2, 0*

*jmpeq &loop\_end*

*moval r0[array], r1*

*addal r0, 2*

*addal r1, 2*

*subal r2, 1*

*jmpal &loop\_begin*

*loop\_end:*

*moval r0, &array*

*moval r1, 20*

*callal &obrada\_zamena*

*moval r0, &array*

*moval r1, 20*

*callal &obrada\_ispis*

*jmpal 0*

***.end***

*Објектни фајл 2:*

***.text***

*.global obrada\_zamena*

*obrada\_zamena:*

*moval r2, r1*

*shlal r2, 1*

*subal r2, 2*

*addal r2, r0*

*dival r1, 2*

*loop\_begin:*

*cmpal r1, 0*

*jmpeq &loop\_end*

*moval r3, r0[0]*

*moval r4, r2[0]*

*moval r0[0], r4*

*moval r2[0], r3*

*addal r0, 2*

*subal r2, 2*

*subal r1, 1*

*jmpal &loop\_begin*

*loop\_end:*

*retal*

***.end***

*Објектни фајл 3:*

***.text***

*.global obrada\_ispis*

*obrada\_ispis:*

*loop\_begin:*

*cmpal r1, 0*

*jmpeq &loop\_end*

*moval r2, r0[0]*

*moval \*65534, r2*

*addal r0, 2*

*subal r1, 1*

*jmpal &loop\_begin*

*loop\_end:*

*retal*

***.end***