Opis metoda i klasa koriscenih u projektu:

Assembler: u njoj su sve glavne metode koje vrse obradu ulaznog asemblerskog koda i ispisuju na izlazu odgovarajuci format, cita ulazni fajl liniju po liniju i procitan sadrzaj prosledju je objektu klase LineParser na dalju obradu

LineParser: klasa koja parsira jednu liniju ulaznog fajla, obradjuje je i prosledju je je nekoj od sledece 3 navedene klase na dalju obradu

DirectiveParser: klasa koja obradjuje direktive asemblerskog fajla

InstructionParser : klasa obradjuje instrukcije asemblerskog fajla

LabelParser: klasa koja obradjuje navedene labele u projektu

RelocaionEntry: klasa koja apstraktuje ulaz u tabeli relokacija za bilo koju sekciju

SectionEntry: klasa koja apstraktuje ulaz u tabeli sekcija za bilo koju sekciju

STEntry: klasa koja apstraktuje ulaz u tabeli simbola

SymbolTable: klasa koja predstavlja tabelu simbola

main: glavna funkcija odakle se ucitava sadrzaj fajla i poziva objekat klase Asembler koji vrsi obradu nad ucitanim ulaznim fajlom

Opis nekih detalja kod sintakse i funkcionalnosti samog asemblera:

Labele , instrukcije i direktive se pisu u zasebnim linijama

Jedino za neposredne apsolutne vrednosti je dozvoljeno 0x tj. pissanje brojeva u hexa obliku

Kod instrukcija ne postoji mogucnost unosenja operanada u obliku izraza

Memorijsko direktno adresiranje ne moze da pocne sa "r"

Kada se koristi registarsko direktno adresiranje za neki operand, onda se npr. r2l i r2 isto kodiraju zbog poslednjeg L/H bita u kodiranju nacina adresiranja datog operanda

Za regindpom je omoguceno samo da stoji w dodatak operaciji instrukcije kada se koristi pomeraj simbola, npr: addw r2,r3[simb] sto znaci da si podrazumevao da offset od simbola simb mora da se koduje na 2B

Za neposredno adresiranje simbola omoguceno je da se offset simbola kodira i na 2B i na 1B tj, omoguceno je i addw r2,simb i addb r2l,simb

Za regindpom nije omoguceno koriscenje r7 jer je to pc

Nije moguce da oba navedena simbola u izrazu kod direktive .word jos nisu definisana jer onda nije moguce ostaviti dobru relokaciju za linker

Za .word nije uzeto da ima razmak posle zareza i nije proveravano za greske tipa 2-simb

Za izracunavanje izraza smeju se koristiti samo jednocifreni brojevi

Nije uzeo u obzir da se u instrukciji moze referenisati simbol iz druge sekcije pa se onda na mestu instrukcije ugradi offset tog simbola a u relokacionu tabelu dodaa relokaciju te sekcije

Testovi:

test1:  
.data   
.byte 1,2,3  
.align 2,1,3  
a:  
.word 1,2,3,4  
  
.section "text  
.global main  
.extern c  
main:  
 mov r0,&a  
 movb r2l,12  
 cmp r0,0  
 jgt kk  
 mov r1[12],a  
 xchg $c,$d  
kk:  
 ret

#tabelaSimbola   
#ime sekcija vr vidlj rb   
.data data 0 1 0   
a data 4 l 1   
.text text 0 1 2   
main text 0 2 3   
kk text 31 2 4   
c ??? -1 2 5   
d ??? -1 2 6   
  
  
  
#.ret.text   
#ofset tip vr [.text]:   
3 1  
10 R\_386\_32 4  
16 R\_386\_32 1  
1a R\_386\_PC32 5  
1d R\_386\_PC32 6  
#.ret.data   
#ofset tip vr [.data]:   
#.ret.bss   
#ofset tip vr [.bss]:   
  
#.text   
1c 20 00 00 00 18 24 00 0c 44 20 00 00 00 ac a0 1f 00 1c 62 0c a0 00 00 0c 8e ff ff 8e ff ff c4   
#.data   
01 02 03 01 01 00 02 00 03 00 04 00   
  
test2:  
.text  
.equ d,20  
.global main  
main:  
 mov r0,a  
 mov r0,&a  
 movw r0,r1[0] #komentar  
 movw r1,\*1234  
  
.end  
  
izlaz2:  
#tabelaSimbola   
#ime sekcija vr vidlj rb   
.text text 0 local 0   
main text 0 global 1   
d text 20 local 2   
  
  
  
#.ret.text   
#ofset tip vr [.text]:   
3 3  
8 3  
#.ret.data   
#ofset tip vr [.data]:   
#.ret.bss   
#ofset tip vr [.bss]:   
  
#.text   
1c 20 a0 14 00 1c 20 00 14 00 1c 20 42 1c 22 a0 d2 04   
#.data