Електротехнички факултет

Универзитет у Београду

Домаћи задатак јун/јул 2015

Системски софтвер

ДВОПРОЛАЗНИ АСЕМБЛЕР

Студент:

Живановић Ђорђе

2013/0033

Професор:

Стојановић Саша

Београд, Јун 2016

# Задатак (25 бодова)

Написати двопролазни асемблер за процесор описан у прилогу. Улаз асемблера је текстуални фајл у складу са синтаксом описаном у наставку. Излаз асемблера треба да буде предметни програм записан у текстуалном фајлу. Формат предметног програма базирати на школској варијанти elf формата (текстуални фајл какав је коришћен у задатку 9 у презентацији V3\_Konstrukcija asemblera.ppt) и предложити измене у формату у складу са потребама циљне архитектуре (нове секције, нови типови записа о релокацијама, додатна поља у постојећим типовима записа и сл.). Приликом генерисања излазног фајла водити се принципима које користи GNU асемблер.

Синтакса асемблера и остали захтјеви:

- у једној линији може бити највише једна команда,

- на почетку сваке линије може да стоји лабела која се завршава двотачком,

- лабела може да стоји и у празној линији и тада је њена вредност једнака адреси прве следеће инструкције,

- симболи могу да се извозе тако што се на почетку фајла наведе директива:

.public <име глобалног симбола>,...

- симболи могу да се увозе тако што се на почетку фајла наведе директива:

.extern <име увеженог симбола>,...

- у једној директиви може да се наведе и више глобалних назива који су одвојени запетама,

- изворни код је подељен у секције:

- .text - секције које садрже машински код,

- .data - секције које садрже иницијализоване податке,

- .bss - секције које садрже неиницијализоване податке,

- свака од секција иза назива може да има и подназив одвојен тачком. Основни назив секције одређује тип секције, док подназив омогућава прављење више различитих секција истог типа (нпр. .text.prvideo, .text.drugideo, .text су 3 различите секције истог типа)

- фајл са изворним кодом се завршава директивом .end. Остатак фајла се одбацује (не преводи се),

- поред мнемоника који су дефинисани треба обезбедити директиве .char, .word, .long, .alignи .skip са истим функционалностима као у GNU асемблеру,

- остатак синтаксе дефинисати по сопственом нахођењу.

# Решење

У директоријуму assembler налази се директоријуми src, bin, tests, који садрже изворни код, извршни код и тестове, као и документацију са описом решења. Код је писан у C++ под оперативним системом Linux Ubuntu 12.04.

У задатку се тражи да се реализује двопролазни асемблер који ће за улазни код који је задат у текстуалној датотеци направити излазну датотеку која ће бити прослеђена на улаз емулатора. Асемблер у првом пролазу пролази кроз цео фајл и формира табелу симбола.

У другом пролазу формирају се објектни код и улази у одговарајућу релокациону табелу симбола тамо где се јавља лабела. На крају се све испише у текстуални фајл задат у комнадној линији у формату код, табела симбола, релокациони записи.

Конструктором за tokenizer који се позива кад се иницијализује assembler, генерише се мапа кључних речи са типом кључне речи, са одговарајућим бинарним симболом ако га има. Непосредне величине могу се задати хексадецимално у формату у стандардном хексадецималном (0x1F), окталном (0234) или децималном формату (234). Директива align равања тренутну адресу тако да буде дељива са 2^param, где је param онај који следи. У .bss једино је дозвољена .skip. У текст секцији су забрањене све директиве изузев align.

Уколико се не наиђе на очекивани формат инструкције, или је у погрешној секцији... Кориснику се јавља грешка и прекида се програм. У std.out има читав log фајл парсирања и линију на којој је неисправан формат инструкције.

Типови реалокација су .long, ldh, ldl, ldclExt, ldchExt. .long се користи за целу 32-битну реч кад није у потпуности одређен израз. ldcl, ldch се користи код ldcl и ldch инструкција. Док ldclExt, и ldchExt код инструкције ldc.

**Коришћене структуре**

* **Assembler** је класа која врши парсирање фајла, прављење објектног кода, табеле симбола и релокационих записа и потом испис истог.
* **AssemblingError** представља класу грешака са пропратним текстом и одговарајућим излазним кодом
* **RellocationTableEntry** представља релокациони записа за одговарајућу релокацију, са свим пољима неопходним за релокацију, offset, section, запис који референцира...
* **RellocationTable** представља табелу релокационих записа за одговарајућу секцију
* **SymbolTableEntry** садржи одговарајући глобални симбол који ће бити видљив у фази линковања и сва поља која га описују (величина, ...)
* **SymbolTable** садржи све глобалне симболе који ће бити видљиви у фази линковања
* **Tokenizer** је класа која се бави парсирањем текста и пријављивањем грешака ако дати формати нису испуњени
* **Main** улазна тачка програма, налази се иницијализација асемблера и одговарајућих пролаза

**Додатне напомене**

Следеће важи:

* све кључне речи се пишу великим словом, као и имена регистара
* директиве .long, .char, .word, i .skip nije moguce koristiti u code sekciji
* у code секцији налазе се само инструкције и .align (на одговорност корисника)
* у data секцији само директиве за алокацију меморије
* код директиве .long подржан је сложен израз са више лабела, или само са једном .extern
* сви симболи који се увозе, извозе морају се навести пре иједне секције
* код директиве .char могуће је написати .char ‘l’, где је l слово
* бројеви су подржани 123, 0x123, 0123, и у .long и негативни и њихово одузимање и сабирање
* код директива .skip и .align садржај се допуњује са нулама,
* код инструкција где је nu садржај, допуњује се са нулама тај део
* померај у релокационом запису је померај инструкције која припада секцији у односу на почетак те секције
* улазна тачка програма је лабела main
* лабеле не смеју имати тачку, не смеју бити кључне речи

**Тестирање**

Програм је преведен коришћењем команде:

g++ -o Assembler Assembler.cpp AssemblingError.cpp Main.cpp RellocationTable.cpp RellocationTableEntry.cpp SymbolTable.cpp SymbolTableEntry.cpp Tokenizer.cpp

И одговарајући извршни фајл је пребачен у одговарајући тест пример и тестиран излаз.

**Тестови**

* **Тест1**

Приказује како се користећи рекурзија исписује број. Приказано је и да је распоред littleEndian у меморији. Да се исправно ажурирају LR, PC, и и да су типови адресирања покривени.

* **Тест2**

Приказује симултани ради тајмера и тастатуре. Ако се брзо буду куцала дугмад на тастатури може се уочити да ће тајмер имати предност. Успут је дат и пример коришћења директиве .align

* **Тест3**

Приказује употребу већине инструкција у асемблеру и како се исправно линкује програм иако је у другом фајлу дефинисана функција за испис броја у бинарном формату.

* **Тест4**

Приказује како се исправно формира објектни фајл, кад су дати сви типови релокационих записа и сви типови директива и на крају и извршни фајл.

* **Тест5**

Приказује како се исправно парсирају сви типови инструкција. Али грешка настаје у linkage делу због недефинисаног симбола a

* **Тест6**

Приказује како се исправно ажурирају битови и ако се направи испремештан садржај у ld скрипти приказује да емулатор добро распореди и изврши програм