Упражнения по
Компютърни архитектури

2023-2024 учебна година

за специалност Информатика,
редовна форма на обучение, първи курс

Конспект 3
за програмиране с NASM и SASM за архитектурата x86-64

Тема
Процедури

Съставил конспекта: Кирил Иванов

Във файла е възможна навигация по съдържание.

В тези конспекти задачите за самостоятелна работа са предназначени за решаване или разглеждане вкъщи без да бъдат отчитани при формирането на оценката по учебния премет. Предвижда се тези задачи да бъдат обсъждани в часовете за упражнения или в консултациите единствено при явно изказване на желание за това от поне един студент.

Акценти по тази тема

Някои принципи на програмирането на асемблерно ниво

▻ Добрият стил на програмиране е необходим за смислено програмиране на асемблерно ниво.

▻ За процедура е необходимо да се описва с коментари, какво прави, какви данни получава, какви данни връща към мястото на извикване и какви специфични особености и странични ефекти има.

▻ Процедурите трябва да избягват всякакви странични ефекти от изпълнението им. В това число флаговете и регистрите с общо предназначение трябва да остават след края на изпълнението на процедурата същите, каквито са били в момента на извикването. Изключение се допуска само за връщане на данни от процедурата към мястото на извикване. Така извикването може да се използва във всеки контекст.

▻ Обикновено променяните флагове и регистри с общо предназначение се съхраняват в стека за да бъдат възстановени преди приключване на процедурата.

➢ Процедурите трябва да работят смислено и при извикване с некоректни параметри (входни данни). Това позволява да бъдат използвани по всякакви начини.

Акценти относно езика NASM

— инструкции `call ...`, `call[...]` и `ret`;

— нелокални и локални етикети; точката пред етикет (например `.Label`) указва, че съответният етикет е продължение на най-близкия преди него нелокален етикет в асемблерния текст;

— всички етикети са достъпни глобално, но локалните могат да изискват уточнение, което започва с нелокален етикет (например `noloc.Label`);

— смесване на части от различни секции в произволен ред;

— използване на `$` за получаване на адрес;

— привличане на (минимални) знания за допълнителен код;

— различни варианти за разклоняване или зацикляне.

Примери и задачи

3.1. Пример

Файл с код: `asm_03_01_demo.asm`

Примерът въвежда три редици от числа, записва ги, точно както физически се разполага масив, и ги извежда.

Всяко въвеждане и изваждане на редица от числа се прави чрез процедура.

Примерът показва:

— дефиниране и извикване на процедура;

— прочитане на входната точка за извиквана процедура от паметта с данни;

— предаване на параметри към процедура чрез регистри;

— избягване на странични ефекти от изпълнението на процедура чрез съхраняване в стека на променяни флагове и регистри и възстановяване на предишните им стойности чрез прочитане от стека преди края на процедурата;

— един вариант за защита на процедурата от извикване с некоректни входни параметри;

— един вариант за коментари, описващи съществената информация за процедура.

3.2. Задача

Файл с примерно решение на задачата: `asm_03_02_task.asm`

Да се напише на асемблерния език NASM с използване на макросите за въвеждане и извеждане на средата SASM приложение, което:

— съдържа изброени цели числа, които разполага в паметта и обработва като масив;

— чрез процедура извежда горните числа по ред на намаляване на адресите им;

— чрез процедура намира максимума на горните числа;

— извежда максимума.

3.3. Задача за самостоятелно решаване

Файл с примерно решение на задачата: `asm_03_03_task.asm`

Да се напише на асемблерния език NASM с използване на макросите за въвеждане и извеждане на средата SASM приложение, което:

а) Чрез процедура въвежда цели числа. Извън процедурата броят на числата трябва да бъде зададен чрез константа в кода, а на самата процедура броят трябва да се подава чрез параметър. Тя трябва да ги записва в аналог на масив, като четните числа записва в ред по нарастване на адресите, започвайки от най-малкия адрес за число, а нечетните записва в ред по намаляване на адресите, започвайки от най-големия адрес, предвиден за число в „масива“.

в) Чрез процедура извежда прочетените числа по ред на нарастване на адресите.

Подсказки за задача 3.3

Допълнителният код на четно число завършва с нула.

Инструкцията **`test N, 1`** изчислява поразрядно „и“ (and) на двата операнда **`N`** и **`1`** без да съхранява резултата, но тя ще получи нула за четно число **`N`** и единица за нечетно число **`N`** и ще модифицира съответно и флага **`ZF`** за нулев резултат.