

*Упражнения по
Компютърни архитектури*

2023-2024 учебна година

*за специалност Информатика,
редовна форма на обучение, първи курс*

*Конспект 2
за програмиране с NASM и SASM за архитектурата x86-64*

*Тема
Макроси на SASM за въвеждане и извеждане.
Цикли за обработка на редици от числа.*

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 предмет. 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

Глобални принципи на архитектурата x86

- Една и съща инструкция може да укаже явно най-много един адрес в паметта.
- В болшинството инструкции с два операнда (например mov, add, sub, логически изчисления и други) и двата операнда трябва да бъдат с еднаква разрядност. Съответно за транслятора е достатъчно да бъде известна разрядността само на единия от тях.
- При наличие на два операнда и получаване на резултат, той се записва в първия операнд.

Два принципа на асемблерните езици за x86

- Типът в асемблерните езици е само разрядност на данните.
- Една и съща инструкция може да работи с данни с различни разрядности. Когато разрядността на операнда не се подразбира, тя трябва да се укаже (с dword, qword и др.).

Акценти относно интегрираната среда SASM

Съобщения за грешки

Съобщенията са неточни и само отчасти назовават причината за регистриране на грешка.

Например:

```
15 section .bss ; секция за резервиране на памет
16 ; (по подразбиране байтовете в нея се нулират)
17 memStr res 100
```

трябва да бъде resq (или resb, resw, resd, ...)

номер на реда, където (когато) е констатирана грешката

съобщението е непълно (само подсказка за вида на грешката)

[11:35:13] Build started...
[11:35:13] Warning! Errors have occurred in the build:
C:\Users\PC\AppData\Local\Temp\SASM\program.asm:17: error: parser: instruction expected
gcc.exe: error: C:\Users\PC\AppData\Local\Temp\SASM\program.o: No such file or directory

Превръщане на код в коментар и обратно

След като се маркира текст

```
6 -5 4 -3 2 -1
6 -5 4 -3 2 -1
-1 , 2 , -3 , 4 , -5 , 6
-1 , 2 , -3 , 4 , -5 , 6
```

```
%include "io64.inc"
section .data
    nums    dq -1, 2, -3, 4, -5, 6
    Len equ ($ - nums) / 8
section .bss
    copy resq Len
section .text
global main
main:
```

с десния бутон на мишката, когато тя сочи маркирания текст, се извиква контекстно меню и от него се избира командата Comment.

```
6 -5 4 -3 2 -1
6 -5 4 -3 2 -1
-1 , 2 , -3 , 4 , -5 , 6
-1 , 2 , -3 , 4 , -5 , 6
```

```
%include "io64.inc"
section .data
    nums    dq -1, 2, -3, 4, -5
    Len equ ($ - nums) / 8
section .bss
    copy resq Len
section .text
global main
main:
```

Comment	Ctrl+Shift+A
Remove comment	Ctrl+Shift+Q
Undo	Ctrl+Z
Redo	Ctrl+Y
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Delete	Del
Select all	Ctrl+A

Тази команда поставя знак „;“ пред първия маркиран знак и в началото на всеки следващ маркиран ред.

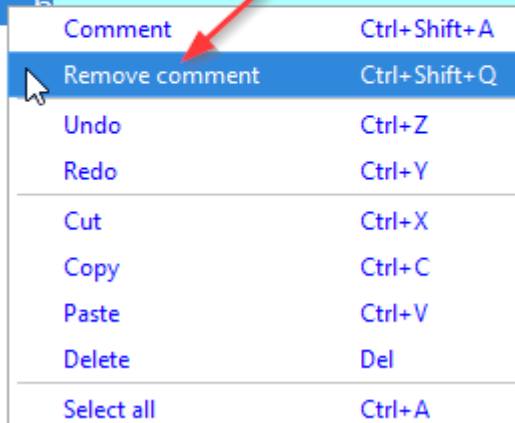
```
    6 -5 ;4 -3 2 -1
;    6 -5 4 -3 2 -1
;   -1 , 2 , -3 , 4 , -5 , 6
;-1 , 2 , -3 , 4 , -5 , 6
```

```
%include "io64.inc"
section .data
    nums    dq -1, 2, -3, 4, -5, 6
    Len equ ($ - nums) / 8
section .bss
    copy resq Len
section .text
global main
main:
```

Обратното преобразование става след маркиране на код коментар с команда Remove comment от контекстното меню.

```
    6 -5 ;4 -3 2 -1
;    6 -5 4 -3 2 -1
;   -1 , 2 , -3 , 4 , -5 , 6
;-1 , 2 , -3 , 4 , -5 , 6
```

```
%include "io64.inc"
section .data
    nums    dq -1, 2, -3, 4, -5,
    Len equ ($ - nums) / 8
section .bss
    copy resq Len
section .text
global main
main:
```



A context menu is displayed over the code. A red arrow points to the 'Remove comment' option, which is highlighted in blue. The menu includes options for Comment, Remove comment, Undo, Redo, Cut, Copy, Paste, Delete, and Select all, each with its corresponding keyboard shortcut.

Comment	Ctrl+Shift+A
Remove comment	Ctrl+Shift+Q
Undo	Ctrl+Z
Redo	Ctrl+Y
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Delete	Del
Select all	Ctrl+A

Така се премахват съответните знакове „;“ за коментар.

```
    6 -5 4 -3 2 -1
    6 -5 4 -3 2 -1
   -1 , 2 , -3 , 4 , -5 , 6
-1 , 2 , -3 , 4 , -5 , 6
```

Акценти относно езика *NASM*

- `%include`;
- `global`;
- инструкции

xor rax, rax

ret

съответни на

global main

...

main: ...;

- `section; .data; .bss; .text;`

— разделяне на една и съща секция на няколко части, разполагани в различни места в асемблерния код;

- разполагане на данни в паметта и заделяне на памет без указване на данни;

- макроси, предоставяни от *SASM*, за въвеждане и извеждане на цели числа и низове;

- константи, въвеждани с `equ`;

— квадратни скоби в асемблерния код и интерпретиране на етикети и константи като адреси и като числа;

— методи на адресация (непосредствена, регистрова, пряка, базова с изместване, индексна с мащабиране и изместване);

- константни изрази;

- цикли с `loop`;

- условни преходи;

- цикли с условни преходи.

Примери и задачи

2.1. Пример

Файл с код: *asm_02_01_demo.asm*

Показва:

- скелет с основните части от приложение;

- деклариране и използване на етикети;

- използване на квадратни скоби в асемблерния език;

- въвеждане на число в 10-ичен и 16-ичен запис и на низ;

- извеждане на число в 10-ичен и 16-ичен запис, на низ и на знакове за нов ред;

— разполагане на конкретни данни и резервиране на памет (*NASM* я нулира по подразбиране) без явно указване на записвани в нея стойности;

- адресации: непосредствена; регистрова; пряка.


2.2. Пример

Файл с код: *asm_02_02_demo.asm*

Показва как след края на стандартния входен поток:

- макросите за прочитане на число връщат нула;
- GET_STRING записва в паметта празен низ;
- макросите за въвеждане продължават да работят без възникване на грешка.

```
] Build started...  
] Built successfully.  
] The program is executing...  
] The program finished normally. Execution time: 0.
```



2.3. Пример

Файл с код: *asm_02_03_demo.asm*

Показва въвеждане и сумиране на две цели числа и извеждане на резултата по различни начини.

2.4. Пример

Файл с код: *asm_02_04_demo.asm*

Показва:

- извеждане на адреси в десетичен запис;
- използване на `eqi`;
- използване на инструкции `mov` (за записване на числото стойност на втория операнд в първия операнд) и `lea` (за записване на адреса на втория операнд, който трябва да бъде зададен чрез адресация в паметта, в първия операнд, който трябва да бъде регистър с разрядност, колкото е разрядността на приложението);
- интерпретиране на адрес като число;
- интерпретиране на константа, декларирана чрез `eqi`, като адрес;
- факта, че няма заделяне на памет за константа, декларирана чрез `eqi`.

2.5. Пример

Файл с код: *asm_02_05_demo.asm*

Примерът решава следната задача:

Да се напише (на асемблерен език) приложение, което въвежда цели числа, чийто брой е определен чрез константа в кода и извежда отначало само сумата им, а след това и цяло твърждение от вида „събираемо+събираемо+...=сума“. При това, всеки от етапите въвеждане, сумиране и извеждане да бъде реализиран чрез отделна част от кода.

Примерът показва:

- разделяне на една и съща секция (`.data`) на различни места в кода;
- следните методи на адресация: непосредствена, регистрова, пряка, базова с изместване; индексна с мащабиране (на индекса) и изместване;
- използване на `eqi` за константа, съществуваща само по време на трансляция;

- използване на „\$“ (за назоваване на адреса непосредствено след последния разположен в паметта машинен код);
- интерпретиране в асемблерния код на една и съща константа или един и същ етикет и като адрес, и като число;
- цикли чрез `loop`;
- цикли чрез условни преходи с инструкциите `jnz` (преход при ненулев резултат при последното изчисление) или `jge` (преход при умаляемо по-голямо или равно от умалителя при предхождащото изваждане).

2.6. Пример

Файл с код: `asm_02_06_demo.asm`

Примерът решава следната задача:

Да се напише (на асемблерен език) приложение, в което:

- а) има декларирана редица от число;
- б) заделено е място, точно колкото за копие на редицата;
- в) създава се копие на редицата от числа;
- г) извеждат се числата от копието по ред на намаляване на адресите им и с по точно един интервал след всяко число;
- д) извеждат се числата от копието по ред на нарастване на адресите им и с подходящ един и същ разделител между всеки две поредни числа.

Примерът акцентира върху начините за програмиране на цикли и достъпа до числата в редицата чрез номер (по номерация от едно), индекс (по индексирание от нула) и адрес.

2.7. Задача за самостоятелно решаване

Файл с примерно решение на задачата: `asm_02_07_task.asm`

Да се напише (на асемблерен език) приложение, което:

- а) Въвежда цели числа, чийто брой е зададен чрез константа в кода.
- б) Извежда прочетените числа по ред, обратен на реда на въвеждането.
- в) Извежда броя на строго положителните прочетени числа.