

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: [tutorcs@163.com](mailto:tutorcs@163.com)

QQ: 749389476

MOVING DATA

<https://tutorcs.com>

SEC204

## 程序代写代做 CS编程辅导

### Overview

- Sections of a program
- Move instruction
- Indexed memory mode



WeChat: cstutorcs

Assignment Project Exam Help

Email: [tutorcs@163.com](mailto:tutorcs@163.com)

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: [tutorcs@163.com](mailto:tutorcs@163.com)

QQ: 749389476

<https://tutorcs.com>

INTRODUCTION

## 程序代写代做 CS编程辅导

# SECTIONS OF A PROGRAM



- **.section .text**
- The text section contains instructions
- Start of the program is defined by the **\_start** label.
  - This indicates the first instruction from which the program should run. If the linker cannot find it, it will produce an error
- **.section .data**
- The data section contains static and global variables (data elements with a static value, variables accessible to all program functions)
- **.section .bss**
- The bss section contains other variables
- We'll talk about the stack and heap later on

```
.section .text
.globl _start
_start:
<Instructions
    here>
```

```
.section .data
<Static and global
variables here>
```

```
.section .bss
<Other variables
    here>
```

WeChat: estutores

Assignment Project Exam Help

Email: [tutorcs@163.com](mailto:tutorcs@163.com)

QQ: 749389476

<https://tutorcs.com>

# 程序代写代做 CS编程辅导

## THE DATA SECTION

To define elements in the data section, you need label and directive



Directive	Data Type
.ascii	Text string
.asciz	Null-terminated text string
.byte	Byte value
.double	Double-precision floating point number
.float	Single-precision floating point number
.int	32-bit integer number
.long	32-bit integer number (same as .int)
.short	16-bit integer number
.single	Single-precision floating point number (same as .float)

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

```
.section .text
.globl _start
_start:
```

```
.section .data
msg:
    .ascii "This is a test"
factors:
    .double 37.45, 45.33, 12.30
height:
    .int 54
length:
    .int 62, 35, 47
```

```
.section .bss
```

## 程序代写代做 CS编程辅导

# THE DATA SECTION

- Each data is placed in memory in the order defined in the data section
- Elements with multiple values are placed in the order listed in the directive



```
.section .data
msg:
    .ascii "This is a test"
factors:
    .double 37.45, 45.33, 12.30
height:
    .int 54
length:
    .int 62, 35, 47
```

Memory  
Addresses

36	
00	
00	54
00	(height)
3E	
00	
00	62
00	(length)
00	
23	
00	
00	35
00	(length)
00	
2F	
00	
00	47
00	(length)
00	

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

## 程序代写代做 CS编程辅导

### STATIC SYMBOLS



- To declare constants (static data) use `.equ` directive
- To reference it, you use the `$ symbol`

```
.section .data
msg:
    .ascii "This is a test"
factors:
    .double 37.45, 45.33, 12.30
height:
    .int 54
length:
    .int 62, 35, 47
.equ factor, 3
.equ LINUX_SYS_CALL, 0x80
```

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

`movl $LINUX_SYS_CALL, %eax`

Moves 0x80 to the eax register

## 程序代写代做 CS编程辅导

# THE BSS SECTION

To define elements in the bss section, you declare raw segments of memory

Directive	Data Type
.comm	Declares a common memory area for data that is not initialised
.lcomm	Declares a local common memory area for data that is not initialised



Create `sizetest1.s` (using code below). Then assemble, and link it to view its size

```
.section .text
.globl _start
_start:
    movl $1, %eax
    movl $0, %ebx
    int $0x80
```

Create `sizetest2.s` (using code below), adding a 10,000-byte buffer. Then assemble, and link it to view its size

```
.section .bss
.lcomm buffer, 10000
.section .text
.globl _start
_start:
    movl $1, %eax
    movl $0, %ebx
    int $0x80
```

Create `sizetest3.s` (using code below), adding a 10,000-byte buffer. Then assemble, and link it to view its size

```
.section .data
buffer:
    .fill 10000
.section .text
.globl _start
_start:
    movl $1, %eax
    movl $0, %ebx
    int $0x80
```

WeChat: cstutorcs

Assignment Project Exam Help

Email: [tutorcs@163.com](mailto:tutorcs@163.com)

QQ: 749389476

<https://tutorcs.com>



## 程序代写代做 CS编程辅导

# MOV INSTRUCTION FORMATS



- MOV source, dest
  - Source and destination can be memory addresses, data values stored in memory, data values defined in the instruction, or registers

WeChat: cstutorcs

- Can define the size of data element to be moved
  - **movl**: l for 32-bit long word value
  - **movw**: w for 16-bit word value
  - **movb**: b for 8-bit byte value

QQ: 749389476

```
movl %eax, %ebx
```

Moves 32-bits %eax to the %ebx register

<https://tutorcs.com>

## 程序代写代做 CS编程辅导

# MOVING DATA



### 1. Between registers

```
movl %eax, %ecx  
movb %al, %bl  
movw %ax, %bx
```

WeChat: cstutorcs

### 2. Between memory and registers

```
movl value, %eax  
movl $10, %eax  
movl %eax, value
```

Assignment Project Exam Help

Email: [tutorcs@163.com](mailto:tutorcs@163.com)

Create file movtest1.s with the following content. Assemble, debug.

```
.section .data  
value:  
    .int 1  
.section .text  
.globl _start:  
nop
```

QQ: 749389476

<https://tutorcs.com>

```
movl value, %ecx  
movl $1, %ebx  
movl $0, %ebx,  
int $0x80
```

## 程序代写代做 CS编程辅导

# INDEXED MEMORY MODE - TABLES



- When you specify memory value on a directive in memory:

```
values:
.int 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60
```

- A sequential series of data values are placed in memory
- Each value occupies one memory unit
- To determine the memory location, we need:
  - A base address
  - An Offset address to add to the base address
  - The size of the data element
  - An index to determine which data element to select

```
base_address (offset_address, index, size)
```

```
memory location = (base_address + offset_address + (index * size))
```

Email: [tutorcs@163.com](mailto:tutorcs@163.com)

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导

## INDEXED MEMORY MODE

- Example: how to access 40 from array values



```
values:
    .int 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60
```

```
movl $2, %edi
movl values(, %edi, 4), %eax
```

WeChat: cstutores

Assignment Project Exam Help

*base\_address (offset\_address, index, size)*

Email: [tutorcs@163.com](mailto:tutorcs@163.com)

*Base\_address: values*

*Offset\_address: (null)*

*Index: %edi (2=third value)*

*Size: 4 (int size)*

QQ: 749389476

<https://tutorcs.com>

## 程序代写代做 CS编程辅导

# INDEXED MEMORY EXAMPLE



Create file movtest3.s with the following content.

```
.section .data
output:
    .asciz "The value is %d\n"
values:
    .int 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60
.section .text
.globl _start:
    nop
    movl $0, %edi
loop:
    movl values(,%edi,4),%eax
    pushl %eax
    pushl $output
    call printf
    addl $8, %esp
    inc %edi
    cmpl $11, %edi
```

...cont...

```
jne loop
movl $0, %ebx
movl $1, %eax
int $0x80
```

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

## 程序代写代做 CS编程辅导

### Indexed memory example

1. Assemble, link, then run the code. What do you see?



```
$as -o movtest3.o movtest.s  
$ld -dynamic-linker /lib/ld-linux.so.2 -lc -o movtest3 movtest3.o  
$./movtest3
```

WeChat: cstutorcs

## Assignment Project Exam Help

Pay particular attention to instruction:

`movl values(, %eax), %eax` Email: [tutorcs@163.com](mailto:tutorcs@163.com)

QQ: 749389476

<https://tutorcs.com>

## 程序代写代做 CS编程辅导

# INDEXED MEMORY POINTERS



- Besides holding data, registers can be used to hold memory addresses
  - When a register holds a memory address, it is referred to as a pointer
  - Accessing data stored in the memory location using the pointer is called indirect addressing

**WeChat: cstutorcs**

- To access the memory location address of a data value, we prepend it with \$

**Assignment Project Exam Help**

```
movl $values, %edi
```

\$values: memory address of values. Moves the memory address of values to the EDI register

**Email: tutors@163.com**

**QQ: 749389476**

**<https://tutorcs.com>**

程序代写代做 CS编程辅导

## INDEXED MEMORY POINTERS



- To use a register as a base use parenthesis

```
movl %ebx, (%edi)
```

Moves the value of EBX to the memory location contained in the EDI register

```
movl %edx, 4(%edi)
```

Moves the value of EDX to the memory location 4 bytes after the location pointed to by the EDI register

```
movl %edx, -4(%edi)
```

Moves the value of EDX to the memory location 4 bytes before the location pointed to by the EDI register

WeChat: cstutorcs

Assignment Project Exam Help

Email: [tutorcs@163.com](mailto:tutorcs@163.com)

QQ: 749389476

<https://tutorcs.com>



## 程序代写代做 CS编程辅导

### POINTERS EXAMPLE



Create file movtest4.s with the following content. Assemble it with gstabs and run it in gdb

```
.section .data
values:
    .int 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60
.section .text
.globl _start:
    nop
    movl values, %eax
    movl $values, %edi
    movl $100, 4(%edi)
    movl $1, %edi
    movl values(, %edi, 4), %ebx
    movl $1, %eax
    int $0x80
```

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

In GDB:

- x/4d &values
- step
- print \$eax
- step
- print/x \$edi
- step
- x/4d &values

程序代写代做 CS编程辅导

## FURTHER READING



- Professional Assembler, chapter 5, pg 91-106

WeChat: cstutorcs

Assignment Project Exam Help

Email: [tutorcs@163.com](mailto:tutorcs@163.com)

QQ: 749389476

<https://tutorcs.com>