

Keene Keannu Kurt C. De Jesus

BSCS3-1

LESSON 26-30 FINAL EXAM

LESSON 26

```
1 ; Close
2 ; Compile with: nasm -f elf close.asm
3 ; Link with (64 bit systems require elf_386 option): ld -m elf_386 close.o -o close
4 ; Run with: ./close
5
6 %include 'functions.asm'
7
8 SECTION .data
9 filename db 'readme.txt', 0h ; the filename to create
10 contents db 'Hello world!', 0h ; the contents to write
11
12 SECTION .bss
13 fileContents resb 255, ; variable to store file contents
14
15 SECTION .text
16 global _start
17
18 _start:
19
20     mov     ecx, 0777o ; Create file from lesson 22
21     mov     ebx, filename
22     mov     eax, 8
23     int     80h
24
25     mov     edx, 12 ; Write contents to file from lesson 23
26     mov     ecx, contents
27     mov     ebx, eax
28     mov     eax, 4
29     int     80h
30
31     mov     ecx, 0 ; Open file from lesson 24
32     mov     ebx, filename
33     mov     eax, 5
34     int     80h
35
36     mov     edx, 12 ; Read file from lesson 25
37     mov     ecx, fileContents
```

```
nasm -f elf close.asm
bash: asm: command not found
ld -m elf_386 close.o -o close
ld: cannot find close.o: No such file or directory
./close
bash: ./close: No such file or directory
Hello world!
bash: Hello: command not found
```

LESSON 27

```
tutorialspoint Online Assembly Compiler
Execute | Beautify | Share | Source Code | Help
```

```
1 ; seek
2 ; Compile with: nasm -f elf seek.asm
3 ; Link with (64 bit systems require elf_386 option): ld -m elf_386 seek.o -o seek
4 ; Run with: ./seek
5
6 %include 'functions.asm'
7
8 SECTION .data
9 filename db 'readme.txt', 0h ; the filename to create
10 contents db '-updated-', 0h ; the contents to write at the start of the file
11
12 SECTION .text
13 global _start
14
15 _start:
16
17     mov     ecx, 1 ; flag for writeonly access mode (O_WRONLY)
18     mov     ebx, filename ; filename of the file to open
19     mov     eax, 5 ; invoke SYS_OPEN (kernel opcode 5)
20     int     80h ; call the kernel
21
22     mov     edx, 2 ; whence argument (SEEK_END)
23     mov     ecx, 0 ; move the cursor 0 bytes
24     mov     ebx, eax ; move the opened file descriptor into EBX
25     mov     eax, 19 ; invoke SYS_LSEEK (kernel opcode 19)
26     int     80h ; call the kernel
27
28     mov     edx, 9 ; number of bytes to write - one for each letter of our contents
29     string ;
30     mov     ecx, contents ; move the memory address of our contents string into ecx
31     mov     ebx, ebx ; move the opened file descriptor into EBX (not required as EBX
32     ; already has the fd)
33     mov     eax, 4 ; invoke SYS_WRITE (kernel opcode 4)
34     int     80h ; call the kernel
35
36     call    quit ; call our quit function
```

```
nasm -f elf seek.asm
ld -m elf_386 seek.o -o seek
./seek
```

LESSON 28

tutorialspointOnline Assembly Compiler

Execute | Results | Show | Source Code | Help

```
1 ; unlink
2 ; compile with: name -f elf_unlink.asm
3 ; link with (64 bit systems require elf_1386 option): ld -m elf_1386 unlink.o -o unlink
4 ; Run with: ./unlink
5
6 atal:
7     push    ebx           ; preserve ebx on the stack to be restored after function runs
8     push    ecx           ; preserve ecx on the stack to be restored after function runs
9     push    edx           ; preserve edx on the stack to be restored after function runs
10    push    esi           ; preserve esi on the stack to be restored after function runs
11    mov     esi, eax       ; move pointer to eax into esi (our number to convert)
12    mov     ecx, 0         ; initialize ecx with decimal value 0
13    mov     ecx, 0         ; initialize ecx with decimal value 0
14
15 .multiplyloop:
16    mov     ebx, ebx       ; resets both lower and upper bytes of ebx to be 0
17    mov     bl, [esi+ecx]  ; move a single byte into ebx register's lower half
18    cmp     bl, 48         ; compare ebx register's lower half value against ascii value 48 (char
                           ; value 0)
19    jl      .finished      ; jump if less than to label finished
20    cmp     bl, 57         ; compare ebx register's lower half value against ascii value 57 (char
                           ; value 9)
21    jg      .finished      ; jump if greater than to label finished
22
23    sub     bl, 48         ; convert ebx register's lower half to decimal representation of ascii
                           ; value
24    add     ecx, ebx       ; add ebx to our integer value in ecx
25    mov     ebx, 10        ; move decimal value 10 into ebx
26    mul     ebx           ; multiply eax by ebx to get place value
27    inc     ecx           ; increment ecx (our counter register)
28    jmp     .multiplyloop  ; continue multiply loop
29
30 .finished:
31    cmp     ecx, 0         ; compare ecx register's value against decimal 0 (our counter register)
32    je      .restore      ; jump if equal to 0 (no integer arguments were passed to atal)
33    mov     ebx, 10        ; move decimal value 10 into ebx
34
```

```
asm -f elf_unlink.asm
ld -m elf_1386 unlink.o -o unlink
ld: cannot find unlink.o: No such file or directory
./unlink
bash: ./unlink: No such file or directory
```

LESSON 29

tutorialspointOnline Assembly Compiler

Execute | Results | Show | Source Code | Help

```
1 ; unlink
2 ; compile with: name -f elf_unlink.asm
3 ; link with (64 bit systems require elf_1386 option): ld -m elf_1386 unlink.o -o unlink
4 ; Run with: ./unlink
5
6 atal:
7     push    ebx           ; preserve ebx on the stack to be restored after function runs
8     push    ecx           ; preserve ecx on the stack to be restored after function runs
9     push    edx           ; preserve edx on the stack to be restored after function runs
10    push    esi           ; preserve esi on the stack to be restored after function runs
11    mov     esi, eax       ; move pointer to eax into esi (our number to convert)
12    mov     ecx, 0         ; initialize ecx with decimal value 0
13    mov     ecx, 0         ; initialize ecx with decimal value 0
14
15 .multiplyloop:
16    mov     ebx, ebx       ; resets both lower and upper bytes of ebx to be 0
17    mov     bl, [esi+ecx]  ; move a single byte into ebx register's lower half
18    cmp     bl, 48         ; compare ebx register's lower half value against ascii value 48 (char
                           ; value 0)
19    jl      .finished      ; jump if less than to label finished
20    cmp     bl, 57         ; compare ebx register's lower half value against ascii value 57 (char
                           ; value 9)
21    jg      .finished      ; jump if greater than to label finished
22
23    sub     bl, 48         ; convert ebx register's lower half to decimal representation of ascii
                           ; value
24    add     ecx, ebx       ; add ebx to our integer value in ecx
25    mov     ebx, 10        ; move decimal value 10 into ebx
26    mul     ebx           ; multiply eax by ebx to get place value
27    inc     ecx           ; increment ecx (our counter register)
28    jmp     .multiplyloop  ; continue multiply loop
29
30 .finished:
31    cmp     ecx, 0         ; compare ecx register's value against decimal 0 (our counter register)
32    je      .restore      ; jump if equal to 0 (no integer arguments were passed to atal)
33    mov     ebx, 10        ; move decimal value 10 into ebx
34
```

```
asm -f elf_socket.asm
ld -m elf_1386 socket.o -o socket
ld: cannot find socket.o: No such file or directory
./socket
bash: ./socket: No such file or directory
3
bash: 3: command not found
```

LESSON 30

tutorialspointOnline Assembly Compiler

Execute | Results | Show | Source Code | Help

```
179
180 .start:
181
182     mov     eax, eax       ; initialize some registers
183     mov     ebx, ebx
184     mov     ecx, ecx
185     mov     esi, esi
186
187 socket:
188
189     push    byte 1         ; create socket from lesson 28
190     push    byte 1
191     push    byte 1
192     mov     ecx, ebp
193     mov     ebx, 1
194     mov     eax, 102
195     int     0x80
196
197 .bind:
198
199     mov     edi, eax       ; move return value of SYS_SOCKETCALL into edi (file descriptor
                           ; for new socket, or -1 on error)
200     push    dword 0xffffffff ; push 0 dec onto the stack 32 ADDRESS (0,0,0,0)
201     push    word 0x0001     ; push 0001 dec onto stack 1677 (reverse byte error)
202     push    word 1         ; push 1 dec onto stack 16787
203     mov     ecx, ebp       ; move address of stack pointer into ecx
204     push    byte 10        ; push 10 dec onto stack (arguments length)
205     push    ecx           ; push the address of arguments onto stack
206     push    edi           ; push the file descriptor onto stack
207     mov     ecx, ebp       ; move address of arguments into ecx
208     mov     ebx, 1         ; invoke subroutine BIND (2)
209     mov     eax, 102       ; invoke SYS_SOCKETCALL (kernel opcode 102)
210     int     0x80          ; call the kernel
211
212 .exit:
213
214     call    quit          ; call our quit function
```

```
asm -f elf_socket.asm
asm -f elf_socket.asm
bash: asm: command not found
ld -m elf_1386 socket.o -o socket
ld: cannot find socket.o: No such file or directory
./socket
bash: ./socket: No such file or directory
```