Introduction to 8086 Assembly

Lecture 15

Macros and the Preprocessor

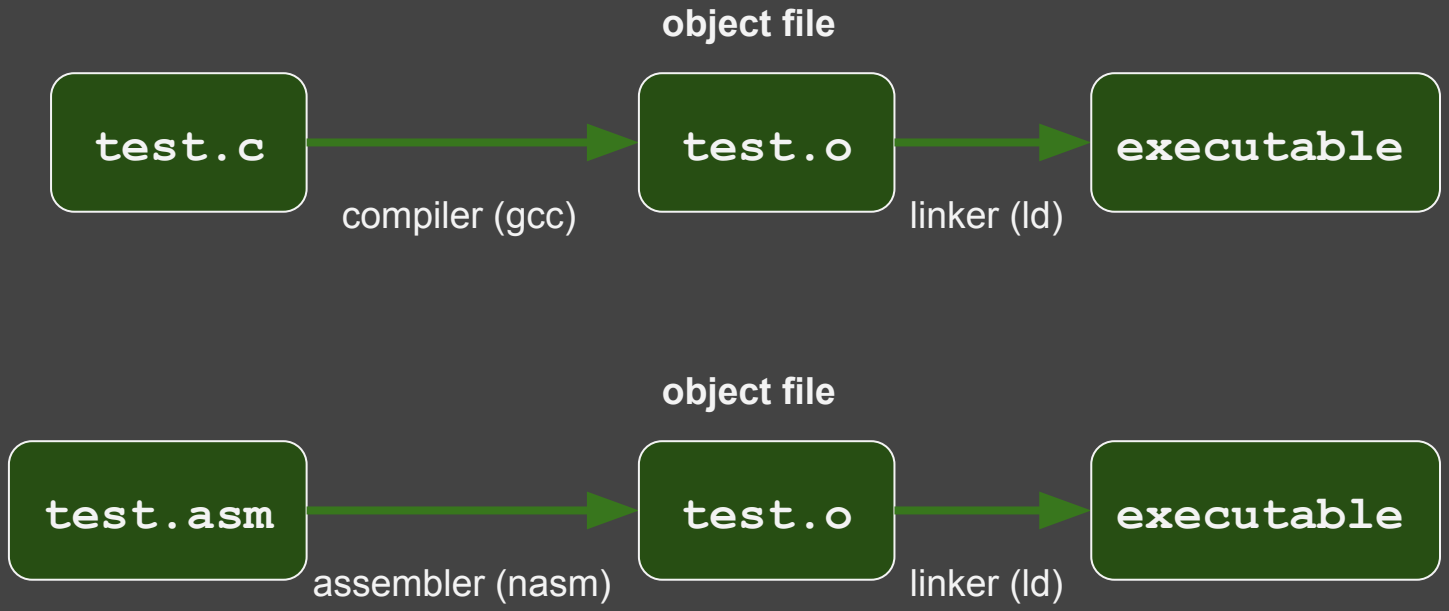
Macros

macro1.asm

```
%define MAX_INT    2147483647
%define MIN_INT    -2147483648
%define MAX_UINT   0xFFFFFFFF

section .text
        global _start

_start:

        mov eax, MAX_INT
        add eax, MIN_INT

        mov ebx, MAX_UINT

        mov     eax,1
        int     0x80
```

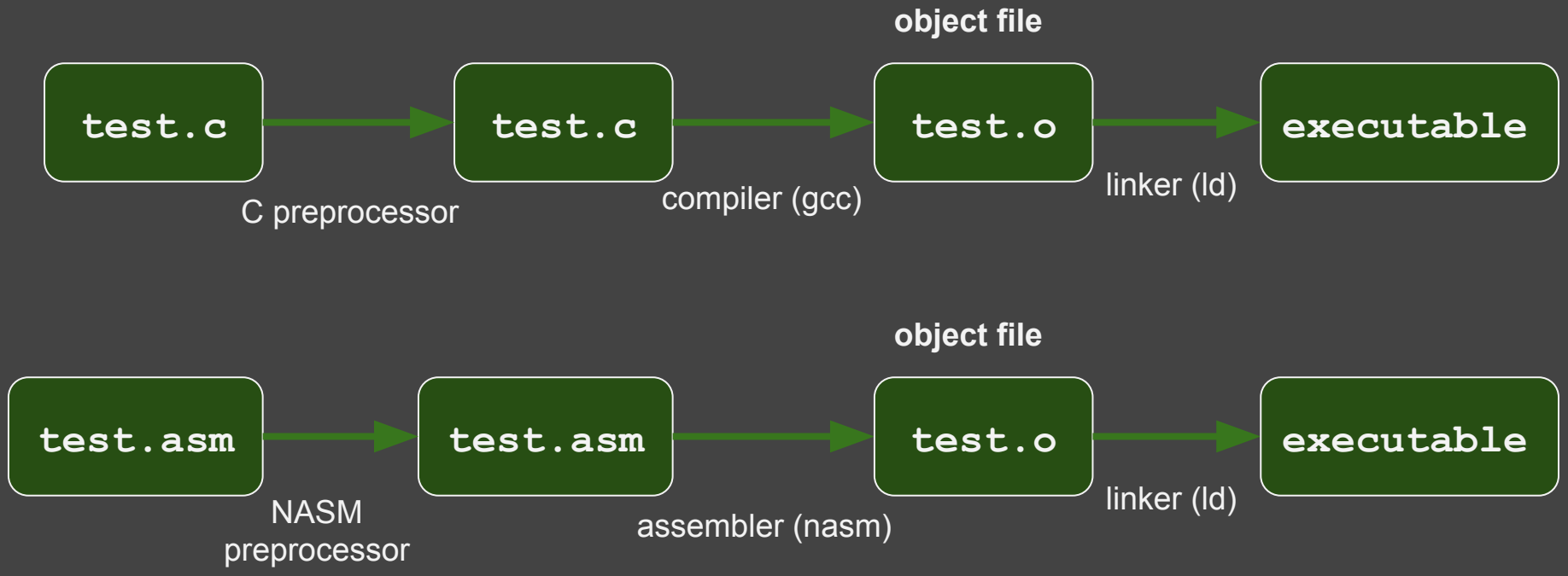
The Preprocessor

K. N. Toosi
University of Technology



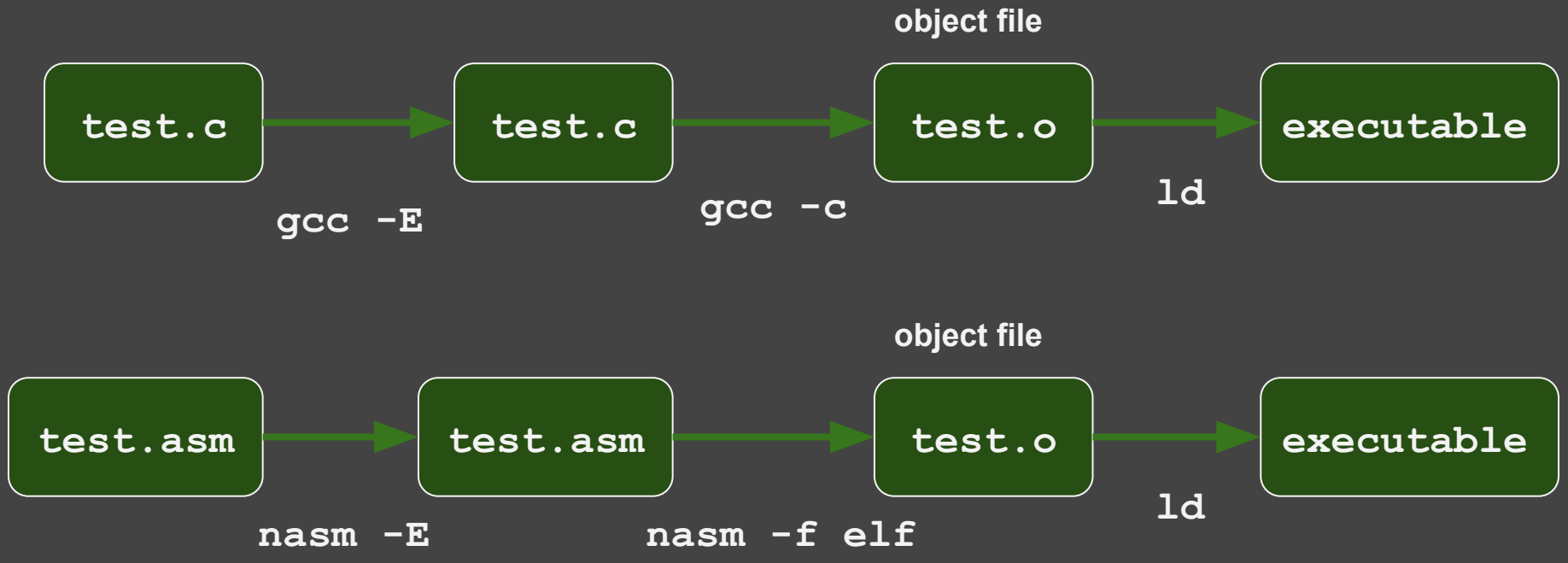
The Preprocessor

K. N. Toosi
University of Technology



The Preprocessor

K. N. Toosi
University of Technology



Macros

macro1.asm

```
%define MAX_INT     2147483647
%define MIN_INT     -2147483648
%define MAX_UINT    0xFFFFFFFF

section .text
        global _start

_start:

        mov eax, MAX_INT
        add eax, MIN_INT

        mov ebx, MAX_UINT

        mov     eax,1
        int     0x80
```

Macros

macro1.asm

```
%define MAX_INT     2147483647
%define MIN_INT     -2147483648
%define MAX_UINT    0xFFFFFFFF

section .text
        global _start

_start:

        mov eax, MAX_INT
        add eax, MIN_INT

        mov ebx, MAX_UINT

        mov     eax,1
        int     0x80
```

```
nasihatkon@kntu:code$ nasm -E macro1.asm
%line 1+1 macro1.asm

%line 5+1 macro1.asm

[section .text]
[global _start]

_start:

 mov eax, 2147483647
 add eax, -2147483648

 mov ebx, 0xFFFFFFFF

 mov eax,1
 int 0x80
```

Macros

macro2.asm

```
%define MAX_INT     Jenabkhan 1234.xyzw neg eax
%define MIN_INT     -2147483648
%define MAX_UINT    0xFFFFFFFF

section .text
        global _start

_start:

        mov eax, MAX_INT
        add eax, MIN_INT

        mov ebx, MAX_UINT

        mov     eax,1
        int     0x80
```

Macros

macro2.asm

```
%define MAX_INT     Jenabkhan 1234.xyzw neg eax
%define MIN_INT     -2147483648
%define MAX_UINT    0xFFFFFFFF

section .text
        global _start

_start:

        mov eax, MAX_INT
        add eax, MIN_INT

        mov ebx, MAX_UINT

        mov     eax,1
        int     0x80
```

```
nasihatkon@kntu:code$ nasm -E macro2.asm
%line 1+1 macro2.asm

%line 5+1 macro2.asm

[section .text]
[global _start]

_start:

 mov eax, Jenabkhan 1234.xyzw neg eax
 add eax, -2147483648

 mov ebx, 0xFFFFFFFF

 mov eax,1
 int 0x80
```

Example: parameters and local variables


```
%include "asm_io.inc"

segment .text
        global asm_main

asm_main:
        pusha

        push 2
        push 8
        call sum

        call print_int
        call print_nl

        popa
        ret
```

```
sum:
        push ebp
        mov ebp, esp

        mov eax, [ebp+8]
        add eax, [ebp+12]

        mov esp, ebp
        pop ebp
        ret 8
```

Example: parameters and local variables


```
%include "asm_io.inc"

segment .text
        global asm_main

asm_main:
        pusha

        push 2
        push 8
        call sum

        call print_int
        call print_nl

        popa
        ret
```

```
sum:
        push ebp
        mov ebp, esp

        mov eax, [ebp+8]
        add eax, [ebp+12]

        mov esp, ebp
        pop ebp
        ret 8
```

macro3.asm

```
;; sum(A,B) {return A+B;}
%define A  [ebp+8]
%define B  [ebp+12]

sum:
        push ebp
        mov ebp, esp

        mov eax, A
        add eax, B

        mov esp, ebp
        pop ebp
        ret 8
```

Redefining Macros

macro3.asm

```
;; sum(A,B) {return A+B;}
%define A  [ebp+8]
%define B  [ebp+12]

sum:
        push ebp
        mov ebp, esp

        mov eax, A
        add eax, B

        mov esp, ebp
        pop ebp
        ret 8

;; calc_ind(N,A,B) {return A*N+B;}
%define N  [ebp+8]
%define A  [ebp+12]
%define B  [ebp+16]

calc_ind:
        push ebp
        mov ebp, esp
```

Macros with arguments


```
%define ref(x)   [x]

segment .text

        mov eax, ref(eax)
```

Macros with arguments


```
%define ref(x)   [x]

segment .text

        mov eax, ref(eax)
```

```
nasihatkon@kntu:code$ nasm -E macroarg1.asm
%line 1+1 macroarg1.asm



[segment .text]

 mov eax, [eax]
```

Macros with arguments

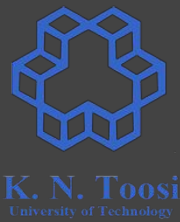

```
%define ref(x)   [x]
%define ref(x,d)   [x+d]
%define ref(x,s,d)   [s*x+d]

segment .text

        mov eax, ref(ebx)

        mov eax, ref(ebx,10)

        mov eax, ref(ebx,4,10)
```

Macros with arguments


```
%define ref(x)    [x]
%define ref(x,d)    [x+d]
%define ref(x,s,d)    [s*x+d]

segment .text

        mov eax, ref(ebx)

        mov eax, ref(ebx,10)

        mov eax, ref(ebx,4,10)
```

```
nasihatkon@kntu:code$ nasm -E macroarg2.asm
%line 1+1 macroarg2.asm

%line 5+1 macroarg2.asm

[segment .text]

 mov eax, [ebx]

 mov eax, [ebx+10]

 mov eax, [4*ebx+10]
```

multi-line Macros

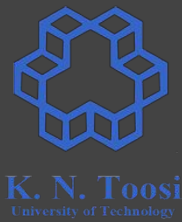



```
%macro macro-name narg

%endmacro
```

multi-line Macros


```
%macro my_enter 1
        push ebp
        mov  ebp, esp
        sub  esp, %1
%endmacro

func:
        my_enter 8
```

multi-line Macros




```
%macro my_enter 1
        push ebp
        mov  ebp, esp
        sub  esp, %1
%endmacro

func:
        my_enter 8
```

```
nasihatkon@kntu:code$ nasm -E macromultiline1.asm
%line 6+1 macromultiline1.asm

func:
 push ebp
%line 8+0 macromultiline1.asm
 mov ebp, esp
 sub esp, 8
%line 9+1 macromultiline1.asm
```

multi-line Macros

```
segment .text

msg1:       db  "Salam Chetori???", 0
newline:    db  10

%macro print_str 2
        pusha

        mov   ebx, 1
        mov   ecx, %1
        mov   edx, %2
        mov   eax, 4
        int   80h

        mov   eax, 4
        mov   ebx, 1
        mov   ecx, newline
        mov   edx, 1
        int   80h

        popa
%endmacro
```

```
%macro  exit  1
        mov ebx, %1
        mov eax, 1
        int 80h
%endmacro

global _start
_start:
        print_str   msg1, 5

        print_str   msg1, 16

        exit 128
```

multi-line Macros

```
segment .text

msg1:       db  "Salam Chetori???", 0
newline:    db  10

%macro print_str 2
        pusha

        mov   ebx, 1
        mov   ecx, %1
        mov   edx, %2
        mov   eax, 4
        int   80h

        mov   eax, 4
        mov   ebx, 1
        mov   ecx, newline
        mov   edx, 1
        int   80h
```

```
%macro  exit  1
        mov ebx, %1
        mov eax, 1
        int 80h
%endmacro

global _start
_start:
        print_str   msg1, 5

        print_str   msg1, 16
```

```
nasihatkon@kntu:code$ nasm -f elf macromultiline2.asm && ld -m elf_i386 macromultiline2.o && ./a.out
Salam
Salam Chetori???
```

Look at Macros in "asm_io.inc"


```
%macro   dump_regs 1
         push      dword %1
         call      sub_dump_regs
%endmacro
```

Look at Macros in "asm_io.inc"


```
%macro   dump_regs 1
         push      dword %1
         call      sub_dump_regs
%endmacro
```

```
%macro   dump_mem 3
         push      dword %1
         push      dword %2
         push      dword %3
         call      sub_dump_mem
%endmacro
```

The %include directive

testinclude.asm

```
%include "myheader.inc"

.text
global _start

_start:
        exit 0
```

myheader.inc

```
extern print_int, print_nl

%macro exit 1
        mov ebx, %1
        mov eax, 1
%endmacro
```

The %include directive

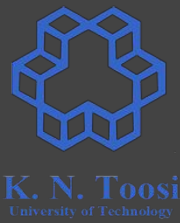
testinclude.asm

```
%include "myheader.inc"

.text
global _start

_start:
        exit 0
```

myheader.inc

```
extern print_int, print_nl

%macro exit 1
        mov ebx, %1
        mov eax, 1
%endmacro
```

The %include directive



testinclude.asm

```
%include "myheader.inc"

.text
global _start

_start:
        exit 0
```

myheader.inc

```
extern print_int, print_nl

%macro exit 1
        mov ebx, %1
        mov eax, 1
%endmacro
```

```
$ nasm -E testinclude.asm

[extern print_int]
[extern print_nl]


.text
[global _start]

_start:
 mov ebx, 0
 mov eax, 1
```

avoid double-inclusion of header files

myheader.inc

```
extern print_int, print_nl

%macro exit 1
        mov ebx, %1
        mov eax, 1
%endmacro
```

myheader2.inc

```
%ifndef _MYHEADER_INC

%define _MYHEADER_INC

extern print_int, print_nl

%macro exit 1
        mov ebx, %1
        mov eax, 1
%endmacro

%endif
```

More on NASM Preprocessor

http://www.nasm.us/doc/nasmdoc4.html

The C preprocessor


```
nasihatkon@kntu:code$ whereis stdio.h
stdio: /usr/include/stdio.h /usr/share/man/man3/stdio.3.gz
nasihatkon@kntu:code$ emacs /usr/include/stdio.h &
[5] 19532
```

Read about C macros

https://www.tutorialspoint.com/cprogramming/c_preprocessors.htm