# Macro Preprocessor

Macro preprocessor helps in processing a macro. It stores the code within the macro body and replaces the macro call with it after making all the necessary modifications. It removes the comments and replaces parameters with the arguments provided.

## **Syntax of defining a macro:**

$startmacro macro\_name [parameter1,parameter2…]

Code block #comment

$endmacro

**Note:** Every parameter defined should start with “$”

## **Syntax of defining a Single line macro:**

$startmacro macro\_name [parameter1,parameter2…] Codeline $endmacro

## **Syntax of defining conditional in a macro:**

$startmacro macro\_name [parameter1,parameter2…]

Code block #comment

&if condition

If code

&endif

&else

Else code

&endelse

&while condition

While code

&endwhile

Code block

$endmacro

## **Programs Accepted:**

$startmacro write $arg1,$arg2

mov eax, 4

mov ebx, 1

mov ecx, $arg1

mov edx, $arg2

int 80h

$endmacro

section .text

global \_start

\_start:

write msg,$arg2=9

mov eax,1

int 80h

**2.**

$startmacro add $arg1=5,$arg2,$arg4

mov eax, $arg1 #move our first number into eax

sub eax, '0'

mov ebx, $arg2 # move our second number into ebx

sub ebx, '0'

add eax, ebx

add eax,'0'

mov [res],ecx # storing the result in res

$startmacro write $arg4

mov eax, 4

mov ebx, 1

mov ecx, eax

mov edx, $arg4

int 80h

$endmacro

$endmacro

section .text

global \_start

\_start:

add ,4,1

mov eax,1

int 80h

**3.**

$startmacro write $arg4

mov eax, 4

mov ebx, 1

mov ecx, eax

mov edx, $arg4

int 80h

$endmacro

$startmacro largest $arg1=5,$arg2,$arg3

mov ecx, $arg1

cmp ecx, $arg2

jg thirdnum

$endmacro

section .text

global \_start

\_start:

largest 5,4,5

write msg

mov eax,1

int 80h

**4.**

$startmacro trywhile $arg5,$arg6

mov eax, $arg5 #to increment

&while $arg5<2

mov ebx, $arg6

add ebx,1

&endwhile

$endmacro

section .text

global \_start

\_start:

trywhile 0,1

mov eax,1

int 80h

**5.**

$startmacro write $arg4

mov eax, 4

mov ebx, 1

mov ecx, eax

mov edx, $arg4

int 80h

$endmacro

$startmacro largest $arg1=5,$arg2,$arg3

mov ecx, $arg1

cmp $arg2

jg thirdnum

$startmacro read $arg1,$arg2 mov eax, $arg1+$arg2 $endmacro #nested single line macro

$endmacro

$startmacro trywhile $arg5

mov eax, $arg5

&while $arg5<2

mov ebx, $arg5

add ebx,1

&endwhile

$endmacro

section .text

global \_start

\_start:

largest 5,4,5

write msg

trywhile 0

mov eax,1

int 80h

**6.**

$startmacro read $arg1,$arg2 mov eax, $arg1+$arg2 $endmacro

$startmacro write $arg4

mov eax, 4

mov ebx, 1

mov ecx, eax

mov edx, $arg4

int 80h

$endmacro

$startmacro largest $arg1=5,$arg2,$arg3

mov ecx, $arg1

cmp $arg2

jg thirdnum

&if $arg1>0 #nested &if and &else

mov eax, $arg1

&if $arg3>0

mov ebx, $arg3

&endif

&else

mov ecx, $arg3

&endelse

&endif

&else

mov eax, $arg2

&endelse

$endmacro

$startmacro trywhile $arg5

mov eax, $arg5

&while $arg5<2 #&while loop

mov ebx, $arg5

add ebx,1

&endwhile

$endmacro

section .text

global \_start

\_start:

largest 5,4,5

write msg

trywhile 0

mov eax,1

int 80h

**7.**

$startmacro read $arg1,$arg2 mov eax, $arg1+$arg2 $endmacro

$startmacro write $arg4

mov eax, 4

mov ebx, 1

mov ecx, eax

mov edx, $arg4

int 80h

$endmacro

$startmacro largest $arg1=5,$arg2,$arg3

mov ecx, $arg1

cmp $arg2

jg thirdnum

read 2,3 #calling macro inside another macro

$endmacro

$startmacro trywhile $arg5

mov eax, $arg5

&while $arg5<2

mov ebx, $arg5

add ebx,1

&endwhile

$endmacro

section .text

global \_start

\_start:

largest 5,4,5

write msg

trywhile 0

mov eax,1

int 80h

## **Features included:**

The features included in this macro preprocessor:

1. **Parameter substitution - positional (with default arguments) and keyword**

This was implemented by storing arguments, parameters and default arguments in separate dictionaries - macroarg, macropar and macrodef respectively - with keys being the name of the macro. Each line was traversed and parameters were substituted according to the requirements

**2. Nested macro implementation**

This was implemented by recursively calling the macro definition function whenever a **$startmacro** was encountered within the body of the parent macro.

**3. Comments removal**

This was implemented by finding the index of **‘#’** in the line and taking the substring before it to store in macro dictionary.

**4. Single line macro implementation**

This was implemented by checking the position of **$endmacro** with respect to **$startmacro** to detect single line definitions and then applying parameter substitution

**5. Conditional**

This was implemented by calling the **checkif** function and **checkwhile** function whenever **&if** and **&while**  was detected respectively. The condition was evaluated which lead to further action.

## **Further Improvements**

1. **&if** cannot be implemented without **&else**.
2. Nested **&while** has not been implemented
3. Condition for **&if** and **&while** only take mathematical and logical expressions with integers
4. Single line macros can only be called in the **beginning** of the line