**Malaviya National Institute Of Technology**



**System Programming Lab**

Mini Project: **DOCUMENTATION**

Multi-purpose MACRO Pre-Processor

Personal details:

* Name-Abhishek Tibrewal
* Id-2016ucp1103
* Email id-2016ucp1103@mnit.ac.in
* Ph. Number-9413163601
* Batch-A1

Professor in-charge:Dr. Arka Prokash Mazumdar

**Design Goal:**

Your designed pre-processor should have important features:

a). Parameter Substitution (positional and parametric, default value)

b). Nested MACRO definitions

c). Comments

d). Single-line/multi-line definitions

e). Conditional Macros, **etc.**

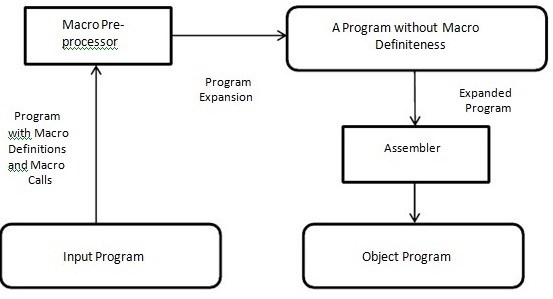
**CONTENTS:-**

* Introduction of MACROS
* MACRO Pre-Processor
* Syntax for used MACROS
* Examples for features included in MACROS

**Introduction of MACROS:**

* Macro instructions are single line abbreviations for group of instructions.
* Using a macro, programmer can define a single instruction to represent block of code.
* Replacement of macro call by corresponding sequence of instructions is called as macro expansion.

**MACRO Pre-Processor:**

****

* A macro processor is a program that copies a stream of text from one place to another, making a systematic set of replacements as it does so.
* Macro processors are often embedded in other programs, such as assemblers and compilers.
* Macro processors have been used for language expansion.

**Syntax for used MACROS:**

* Macro definition:

!!!M\_START ..macro\_name.. (parameters)

Statements in macro definition for expantion and parameter substitution

!!!M\_FINISH

* Macro starts with **!!!M\_START**
* Macro ends with **!!!M\_FINISH**
* Macro name is any name except keywords
* Macro call:

..macro\_name.. (arg1,arg2,arg3,………..)

* Parameters:
* &arg1&=1
* Parameter name starts with ‘&’ and ends with ‘&’.
* All parameters are assigned default value 1;
* Parameters can be passed in any order.
* Comments:

<\*you are in comment\*>

* Comment starts from ‘<\*’
* Comment ends with ‘\*>’
* Nested macro definition:  
  !!!M\_START ..macro1.. (parameters)

Statements

!!!M\_START ..macro2.. (parameters)

Statements

!!!M\_FINISH

!!!M\_FINISH

* The inner macro cannot be called before expanding of outer macro.
* If outer macro if called then only we can call nested(inner) macros separately.
* Single line definition:

!!!M\_STARTS ..macro1.. (&a1&=1) int: db '%d',&a1&,&a2& !!!M\_FINISHS

* Starts with !!!M\_STARTS
* Ends with !!!M\_FINISHS
* Conditional macro(if-else):

!!!M\_START ..condition.. (&a1&=1,&a2&=1)

IF..(&a2& L &a4&)

statements

ELSE..

statements

END\_IFF

!!!M\_FINISH

* If block starts with IF..
* Else block starts with ELSE..
* If block ends with END\_IFF
* E for =
* N for !=
* G for >
* L for <
* Conditional macro(while loop):

!!!M\_START ..function.. (&a1&=1,&a2&=1)

WHILE..(&a1& L &a2&)

mov ecx,&a5&

INCR &a1&

ENDW..

!!!M\_FINISH

* While loop starts with WHILE..
* INCR is for increament.
* DECR is for decreament.
* Loop ends with ENDW..
* N for !=
* G for >
* L for <

**Examples for features included in MACROS:**

**a). Parameter Substitution**

!!!M\_START ..function.. (&a1&=1,&a2&=1,&a3&=1)

<\*you are in comment\*>

extern printf

extern atoi

!!!M\_FINISH

!!!M\_START ..macro1.. (&a1&=1,&a2&=1,&a3&=1,&a4&=1)

<\*for data section\*>

int: db '%d',&a1&,&a2&

!!!M\_FINISH

..function.. ()

SECTION .data

..macro1.. (10,0)

SECTION .bss

i RESB 32

n RESB 32

S RESB 32

SECTION .text

global main

main:

push ebp

mov ebp,esp

mov ebx,dword[esp+12]

mov ecx,[ebx+4]

pop ebp

ret

**b). Nested MACRO definitions**

!!!M\_START ..macro2.. (&a1&=1,&a2&=1,&a3&=1,&a4&=1,&a5&=1)

<\*nested macro\*>

iteration:

mov ebx,[n]

cmp ebx,1

je end

add eax,ecx

mov ecx,eax

push eax

push int

call &a6

jmp iteration

!!!M\_START ..macro3.. (&a1&=1,&a2&=1,&a3&=1)

end:

mov esp,ebp

!!!M\_START ..macro4.. (&a1&=1)

pop ebp

ret

!!!M\_FINISH

!!!M\_FINISH

!!!M\_FINISH

SECTION .text

global main

main:

push ebp

mov ebp,esp

..macro2.. ()

**c). Comments**

!!!M\_START ..function.. (&a1&=1,&a2&=1,&a3&=1,&a4&=1)

<\*you are in comment\*>

<\*you are in comment\*>

extern printf

extern atoi

..function.. ()

SECTION .bss

i RESB 32

n RESB 32

S RESB 32

**d). Single line macro definition**

!!!M\_STARTS ..macro1.. (&a1&=1) int: db '%d',&a1&,&a2& !!!M\_FINISHS

extern printf

extern atoi

SECTION .data

..macro1.. (10,0)

SECTION .bss

i RESB 32

n RESB 32

S RESB 32

**e-1). Conditional macro(if-else)**

!!!M\_START ..condition.. (&a1&=1,&a2&=1,&a3&=1)

IF..(&a2& L &a3&)

iteration:

mov ebx,[n]

cmp ebx,1

je end

add eax,ecx

mov ecx,eax

push eax

push int

call &a6&

jmp iteration

ELSE..

end:

mov esp,ebp

pop ebp

ret

END\_IFF

!!!M\_FINISH

SECTION .text

global main

main:

push ebp

mov ebp,esp

..condition.. (1,6,3)

**e-2). Conditional macro(while loop)**

!!!M\_START ..function.. (&a1&=1,&a2&=1,&a3&=1)

WHILE..(&a1& L &a2&)

mov ecx,&a3&

INCR &a1&

ENDW..

!!!M\_FINISH

SECTION .text

global main

main:

push ebp

mov ebp,esp

mov ebx,dword[esp+12]

mov ecx,[ebx+4]

push ecx

call atoi

mov [n],eax

mov eax,1

mov [i],eax

..function.. (0,4,0)

mov eax,0

mov [S],eax

push eax

push int