Malaviya National Institute Of Technology

System Programming Lab

DOCUMENTATION

For **SOPHOS**

A Multi-Purpose Macro Pre-processor

Developed By :-

Adarsh Kumar Jain

Arpit Kumawat

2015ucp1547@mnit.ac.in

2015ucp1524@mnit.ac.in

What are Macros ?

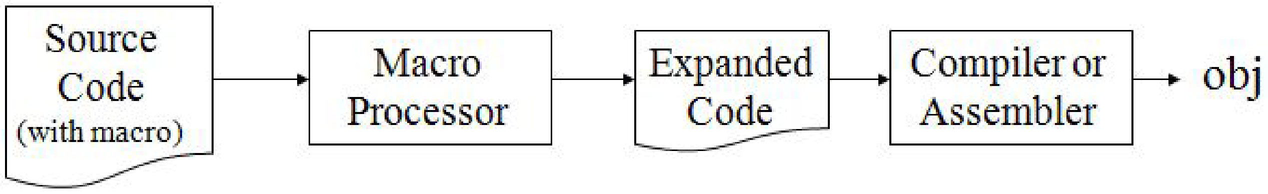
- ✓ A **macro** (which stands for "macroinstruction") is used to make certain tasks less repetitive by representing a complicated sequence of commands or statements into a **shorthand notation.**
- ✓ Thus they allow a developer to **re-use code** and are for notational convenience.

Note that a **macro is not the same as a** function : functions require special instructions and computational overhead to safely pass arguments and return values.

What is a Macro Pre-processor ?

- ✓ Preprocessor is just a **tool that allows us to use macros** in a program and instructs the compiler to do required pre-processing before the actual compilation.

- ✓ It replaces each macro invocation (call) with the corresponding sequence of statements (expansion) .



Why SOPHOS ?

SOPHOS is a **general purpose** macro pre-processor in the sense that is not tied to or integrated with a particular language or piece of software.

It is **developed in python** language and it is suitable for both low level language (like NASM) and high level languages (like Python and C).

Features of SOPHOS :-

- ✓ Single line and multi line macro definitions

- ✓ Has its own single line and multi line comments for the convenience of programmer
- ✓ Allows nested macro definitions and calls
- ✓ Has conditional macro definitions (if else clauses)
- ✓ Allows macro overloading
- ✓ Is suitable for both high and low level languages

Syntax for Using macros in SOPHOS

Macro definitions

Single Line Macro Definition :

Syntax :

$macd   <macro-name>   <expansion-statements>

Or

$macd <macro-name> (<parameters>) <expansion-stmts>

Multi Line Macro Definition :

Syntax : *$macd . . .*

<macro-name> (<parameters>)

<expansion-statements>

$$

macro-name : Valid identifier name as in most of programming language.

parameters : Valid identifier name as in most of programming language, may take default values (like tax=10).

expansion-stmts : Any statements that user needs to replace using this macro.

Note : parameters are optional in both the cases.

Including Comments

Single Line Comments : using the symbols ***'- - '***

Syntax : **- -** *this is a single line comment*

Multi Line Comments : using the symbols ***'<# #>'***

Syntax : *<# this is a*

multi line comment #>

Conditional Macros

Syntax :

```
$if    < expression / macro >
        < expansion-statements-1>
$elif    < expression / macro >
        < expansion-statements-2>
$else    < expression / macro >
        < expansion-statements-3>
$end
```

expression : Expression to test.

macro : To set statements by checking if a macro exists or not.

Preprocessing a file using SOPHOS

Link to download SOPHOS

SOPHOS can be downloaded easily from our GitHub repository :

https://github.com/Adarsh-sophos/MACRO-Pre-Processor

Preprocessing a file

1. Extract the downloaded zip file
2. Ensure that you have python installed as it runs on python
3. Place your code file containing macros defined in SOPHOS into example programs directory present in the extracted directory
4. Now run macro.py file
5. You will be prompted to enter file name
6. Enter your file name along with its extension (eg. code.txt) and press enter
7. Your file will be preprocessed and the output file will be saved by the name “<your_file_name>o.txt” in the example programs folder
8. Run it and enjoy !!

}

Some sample macro examples depicting specific features

Single line macro :

1. Example for Python

```
$macd   A 10
$macd  B(a) print(“The value of parameter is “ + str(a) )
print(“The two defined macros will be called here “)
B(20)
print(“Macro A has value “ + str(A) )
```

2. Example for C

```
#include<stdio.h>
$macd   MAX(x=0, y) x>y?x:y;
$macd  HELLO “lets greet our user !! Hello user”
void main()
{
        printf(HELLO);
        int Maximum=MAX(10,20)
}
```

3. Example for NASM

```
$macd    stmfora  db "a=%d", 10, 0


SECTION .data
a: dd 6
stm: stmfora


SECTION .text
extern printf
global main
main:
        push ebp
        mov ebp,esp
        push dword [a]
        push stm
        call printf
        add esp,8
        mov esp, ebp
        pop ebp
        ret
```

Multi Line Macro :

4. Example for Python

```
$macd ...
        SUM1(a,b,c=5)
                print(“sum is”)
                x=a + b + c
                print(x)
$$
print(“Program to calculate sum of three numbers “ )
SUM1(10 , 27 )
```

5. Example for C

```
#include<stdio.h>
$macd ...
        SUM(c, k, a=56, b=12)
                printf("The sum is ");
                int x = a+b+c+k;
$$
void main()
{
        SUM(10,20,30)
        printf("%d", x);
}
```

6. Example for NASM

```
$macd ...
        stmfora( )
                a: dd 6
                stm : db "a=%d", 10, 0
$$
SECTION .data
stmfora
SECTION .text
extern printf
global main

main:
        push ebp
        mov ebp,esp
        push dword [a]
        push stm
        call printf
        add esp,8
        mov esp, ebp
        pop ebp
        ret
```

Single Line And Multi Line Comments :

7. Example for Python

```
<#    this example uses
        A multi line
        comment   #>
$macd  A 10
print(“Macro  has value “+ str(A) )
```

8. Example for C

```
#include<stdio.h>
--Using single line macro to comment some definitions
 --$macd F 60
$macd H 80
$macd MIN 10

Void main()
{
        printf(“macro h is %d“,H );
        printf(“macro min is %d“,MIN );
}
```

9. Example for NASM

```
$macd  ...
        stmt(  )
                <#      a: dd 6
                        commented so not replaced   #>
                stm : db "a=%d", 10, 0
$$
SECTION .data
stmt(  )
SECTION .text
extern printf
global main
main:
        push ebp
        mov ebp,esp
        push stm
        call printf
        add esp,8
        mov esp, ebp
        pop ebp
        ret
```

Conditional macro :

10. Example For Python

```
$macd e 50
--$macd d 60

$if d
print(“d  is defined “)

$elif e
print(“e  is defined “)

$else
Print(“Both d and e are not defined “)
$end
```

11. Example For C

```
#include<stdio.h>

$macd   G 10
$macd   H 200
$macd   E 30
```

```
$macd   D 0

void main()
{
        $if D
                $if H
                        printf(“h and d are defined");
                $else
                        printf(“d is defined");
                $end
        $elif E
                printf(“e is defined”);
        $else
                printf(“d and e are not defined “);
        $end
}
```

12. Example For NASM

```
$macd   G test

SECTION .data
        $if G
                Greet:  db “Hello !!”,10,0
        $else
                Greet:  db  “No greeting”,10,0
        $end
```

```
SECTION .text
extern printf
global main

main:
        push ebp
        mov ebp,esp
        push stm
        call printf
        add esp,4
        mov esp, ebp
        pop ebp
        ret
```

<u>Macro Overloading :</u>

This macro preprocessor allows macro overloading that is same name macros can be created if they have different number of parameters. Thus same name macro can have multiple definitions.

13. Example for macro overloading (Example For C)

```
<#      Here same macro ‘sum’ is defined
        multiple times      with different number of parameters       #>
```

```
$macd ...
SUM(a,b,c=1,d=2,e=3)
        printf(“This sum has 5 parameters : “);
        printf(“%d %d %d %d %d”, a,b,c,d,e);
$$

$macd ...
SUM(a=10)
        printf(“This sum has no parameters”);
$$

$macd ...
SUM(a,b,c)
        printf(“This sum has 3 parameters :”);
        printf(“”%d %d %d”, a,b,c);
$$

void main()
{
        SUM(10,20,30,40,50)
        SUM(10,20,30)
        SUM(50)
}
```

Nested Macro Definition And Calls

14. Calling a macro inside definition of another (Python)

```
$macd  SUM(a,b,c) sum3( b, c) print(a)


$macd ...
sum3( x , y )
        total=x+y
        print(total)
$$

SUM(5,10,20)
```

15. Defining a macro inside another macro (Example for C)

```
$macd ...
        SUM(a,b,c=5)
        printf("the sum is ");
                $macd ...
                        SUM12(a,b,c,d)
                        printf("This is a nested macro definition ");
                        int x=a + b + c+d;
                $$
        int x=a+b+c;
$$

SUM12 (5, 10, 2, 6)
```

16. Another example for nested call (swap)

```
$macd ...
        SWAP(a,b,c,d,e,f,g,h)
                SWAP(a,b,c,d)
                SWAP(e,f,g,h)
$$

$macd ...
        SWAP(a,b,c,d)
                SWAP(a,b)
                SWAP(c,d)
$$

$macd ...
        SWAP(a,b)
                b,a
$$

SWAP(1,2,3,4,5,6,7,8)
```

17. Example for extended conditional statements

```
#include<stdio.h>
$macd A 10
$macd B 20
--$macd C 30
```

```
$macd D 40
--$macd E 50
--$macd F 60
--$macd G 70
$macd H 80
$macd I 90
$macd J 100

void main(  )
{
        $if A
                $if D
                        $if H
                                $if I
                                        $if J
                                                printf("j is defined");
                                                printf("i am in j");
                                        $end
                                $end
                        $end
                $elif E
                        printf("e is defined");
                        printf("i am e");
                $else
                        printf("i am in else of first if");
                $end
        $elif C
                $if F
                        printf("i am in f");
                        printf(f is defined);
```

```
            $else
                    printf("nothing is defined");
                    printf("i am in 2nd elif condition");
            $end
    $else
            $if G
                    printf("i am in g");
            $else
                    printf("do nothing");
            $end
    $end
}
```

18. Expr

```
    $macd ...
    ADD(x=2, y=3)
            int a = x;
            int b = y;
            printf("%d", x*2/y+x+x*y);
    $$

    int main()
    {
            ADD(1)
            pritnf("a=%d",a);
            return 0;
    }
```