SYSTEM PROGRAMMING LAB

CST 210

MACRO

PREPROCESSOR DOCUMENT

BY: ANMOLA KUMARI

2016UCP1422

B1

(under the guidance of professor Arka Prokash Mazumdar)

What are macros?

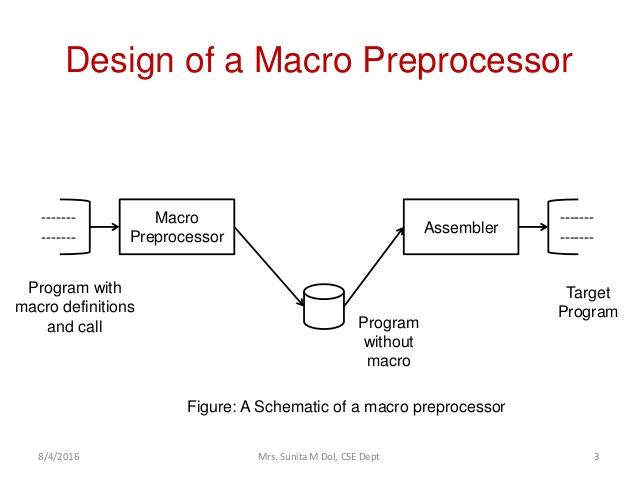
* A *macro* is a fragment of code which has been given a name. Whenever the name is used, it is replaced by the contents of the macro.
* It consists of name, a set of formal parameters and a body of code
* The use of name with the set of actual parameter is replaced by code generated by its body

Macro differ from subroutines:

* Macro name use lead to expansion whereas subroutine name lead to execution
* Macro require greater size with more efficiency than subroutine.

Macro processor

* A macro processor is a program that reads a file (or files) and scans them for certain keywords. When a keyword is found, it is replaced by some text. The keyword/text combination is called a macro.



* The [preprocessor](https://en.wikipedia.org/wiki/Preprocessor) is a part of the compiler which performs preliminary operations (conditionally compiling code, including files etc...) to the code before the compiler sees it. These transformations are lexical, meaning that the output of the preprocessor is still text.

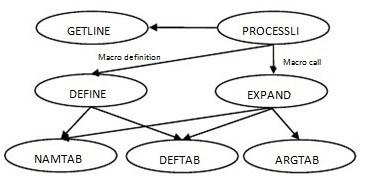
One pass macro processor

* A one-pass macro processor is another design option available for macro processing. The restriction in working with one-pass macro processors is that they strictly require the definition of a macro to appear always before any statements that invoke that macro in the program.
* The important data structures required in a one-pass macro processor are:

**DEFTAB (Definition Table):** store the macro definition including macro prototype and macro body.

**NAMTAB (Name Table):**used for storing macros names

**ARGTAB (Argument Table)**: maintains arguments according to their positions i in the argument list.



One-pass macro processor schematic

Project of Macro processor constitutes:

* This is suitable for both NASM and GAS
* allow user to do single macro
* multi line macro with parameters
* passing arguments into the macro
* removing single line
* removing multi line comment
* nested definition and macro call
* conditional macro

syntax for using macro:

* macro definition



Example:



Hello ………. Name of the macro

MACRO ……… start of definition of macro

LDA &a1 …..... instruction or body of macro

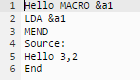
MEND ……… macro end

* calling macro in main body and its expansion:

Macro expansion:

* replacement of macro call by the corresponding sequence of instructions .

macro definition and call:



After expansion:

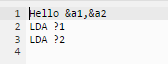


All the three table generated are:

Nametab:



Deftab:



Argtab:

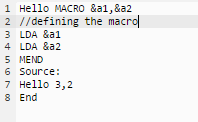


Process of implementation:

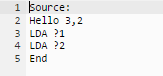
* Input file is taken as main.txt and is read line by line to fill nametab ,deftab, argtab table.
* While reading if “MACRO” word is found, it prints the name of macro in nametab.
* For filling deftab, if any comment is there, it is ignored and loop is run till “MEND” is found. Body of macro is printed in deftab and “&” sign is replaced by “?” .
* From the source part, if the name is encountered the there must be arguments associated with the name of macro, all the parameters passed are printed in argtab
* Output file contains expansion of the macro in source part with the formal parameters being replaced actual parameter.
* Removing comments:

“//” is being used for the commenting.

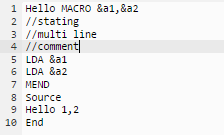
Input file with single line comment:



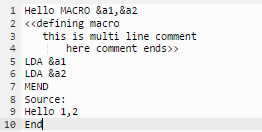
Output file without the comment:



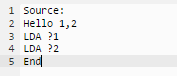
Input file with multi line comment:



or



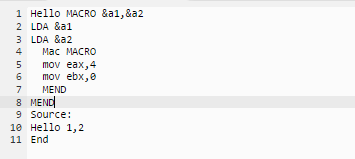
Output file without multi line:

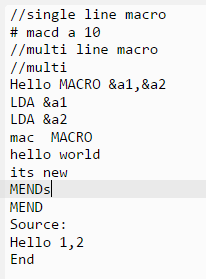


Nested macro: definition of one macro within the other:

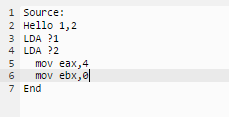
* Macro is defined and expanded within the another macro.
* Inner macro can’t be expanded till the outer macro is defined.
* If outer macro is called in main part then the inner macro is also expanded.

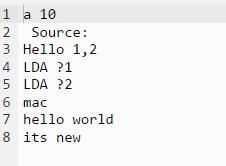
Input file for nested macro:





Output file for nested macro:





PART 2:

Single line macro: defining a macro in a single line

#<macro\_name> variable data

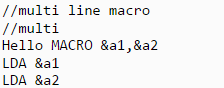


Multi line macro:

<macro\_name>MACRO <arguments>

definition

MEND



* Conditional macro: macro is expanded after taking into consideration some conditions.

Syntax:

<macro\_name>MACRO <arguments>

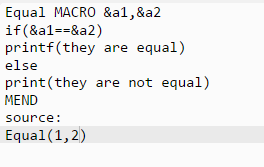
If(<condition>)

<Statement1>

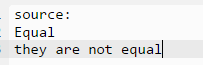
Else

<statement2>

Eg:



output file generated:



code to run :

chmod a+x filename.py

python3 filename.py

