



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
 (Empowered Autonomous Institute Affiliated to University of Mumbai)
 [Knowledge is Nectar]

Department of Computer Engineering

Course - System Programming and Compiler Construction (SPCC)

Aim	Design a two pass Macro Processor
Objective	Implement macros for assembly language instructions to improve code readability, efficiency, and maintainability in a simulated environment.
Theory	<p style="text-align: center;">MACRO PREPROCESSOR</p> <p>Macro</p> <ul style="list-style-type: none"> • A macro (or macro instruction) <ul style="list-style-type: none"> ○ It is simply a notational convenience for the programmer. ○ It allows the programmer to write shorthand version of a program ○ It represents a commonly used group of statements in the source program. • For example: <ul style="list-style-type: none"> ○ Suppose a program needs to add two numbers frequently. This requires a sequence of instructions. We can define and use a macro called SUM, to represent this sequence of instructions. <pre style="text-align: center;">SUM MACRO &X,&Y LDA &X MOV B LDA &Y ADD B MEND</pre> <p>Macro Preprocessor</p> <ul style="list-style-type: none"> • The macro pre-processor(or macro processor) is a system software which replaces each macro instruction with the corresponding group of source language statements. • This operation is called expanding the macro. • It does not concern the meaning of the involved statements during macro expansion. • The design of a macro processor generally is machine independent. <div style="text-align: center;"> <pre> graph LR A[Source Code (with macro)] --> B[Macro Processor] B --> C[Expanded Code (without macro)] C --> D[Compiler or Assembler] D --> E[object program] </pre> </div> <p style="text-align: center;">BASIC MACRO PROCESSOR FUNCTIONS</p> <p>The fundamental functions common to all macro processors are: (Code to remember - DIE)</p> <ul style="list-style-type: none"> • Macro Definition • Macro Invocation • Macro Expansion <p>Macro Definition</p> <ul style="list-style-type: none"> • Macro definitions are typically located at the start of a program. • A macro definition is enclosed between a macro header statement(MACRO) and a macro end statement(MEND)



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to University of Mumbai)
[Knowledge is Nectar]

Department of Computer Engineering

- Format of macro definition
macroname MACRO parameters
:
body
:
MEND
- A macro definition consist of macro prototype statement and body of macro.
- A macro prototype statement declares the name of a macro and its parameters. It has following format:

macroname MACRO parameters

where *macroname* indicates the name of macro, *MACRO* indicates the beginning of macro definition and *parameters* indicates the list of formal parameters. *parameters* is of the form ¶meter1, ¶meter2,... Each parameter begins with '&'.

Whenever we use the term macro prototype it simply means the macro name along with its parameters.

- Body of macro consist of statements that will generated as the expansion of macro.
- Consider the following macro definition:

```
SUM MACRO &X,&Y
LDA &X
MOV B
LDA &Y
ADD B
MEND
```

Here, the macro named SUM is used to find the sum of two variables passed to it.

Macro Invocation(or Macro Call)

- A macro invocation statement (a macro call) gives the name of the macro instruction being invoked and the arguments to be used in expanding the macro.
- The format of macro invocation
macroname p1, p2,...pn
- The above defined macro can be called as SUM P,Q

Macro Expansion

- Each macro invocation statement will be expanded into the statements that form the body of the macro.
- Arguments from the macro invocation are substituted for the parameters in the macro prototype.
- The arguments and parameters are associated with one another according to their positions. The first argument in the macro invocation corresponds to the first parameter in the macro prototype, etc.
- Comment lines within the macro body have been deleted, but comments on individual statements have been retained. Macro invocation statement itself has been included as a comment line.
- Consider the example for macro expansion on next page:

In this example, the macro named SUM is defined at the start of the program. This macro is invoked with the macro call SUM P,Q and the macro is expanded as



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to University of Mumbai)
[Knowledge is Nectar]

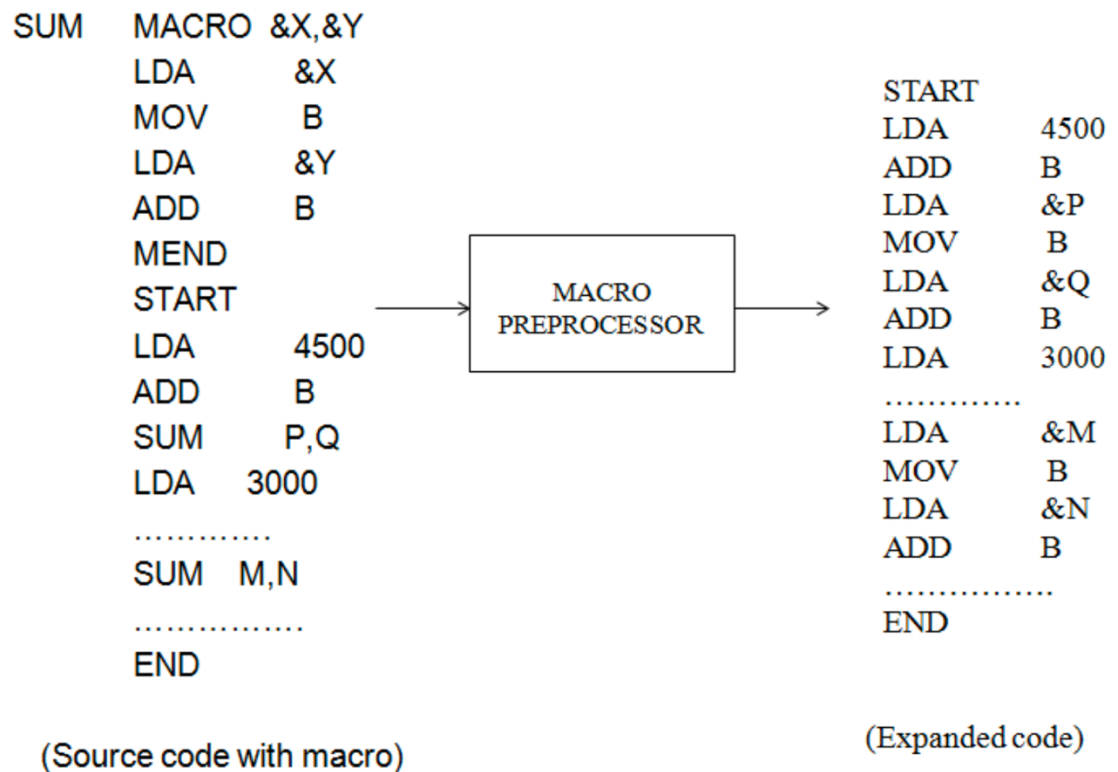
Department of Computer Engineering

```
LDA &P
MOV B
LDA &Q
ADD B
MEND
```

Again the same macro is invoked with the macro call SUM M,N and the macro is expanded as

```
LDA &M
MOV B
LDA &N
ADD B
MEND
```

Figure: Example for macro expansion



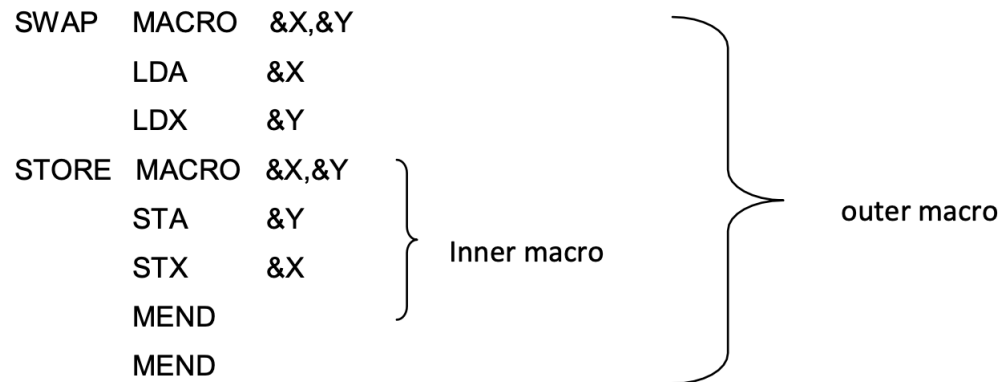
Two pass macro processor

- It is easy to design a two-pass macro processor in which all macro definitions are processed during the first pass and all macro invocation statements are expanded during second pass.
- Such a two pass macro processor cannot handle **nested macro definitions**. Nested macros are macros in which definition of one macro contains definition of other macros.
- Consider the macro definition example given below, which is used to swap two numbers. The macro named SWAP defines another macro named STORE inside it. These type of macro are called nested macros.



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to University of Mumbai)
[Knowledge is Nectar]

Department of Computer Engineering



One pass macro processor

- A one-pass macro processor uses only one pass for processing macro definitions and macro expansions.
- It can handle nested macro definitions.
- To implement one pass macro processor, the definition of a macro must appear in the source program before any statements that invoke that macro.

Data Structures involved in the design of one pass macro processor

- There are 3 main data structures involved in the design of one pass macro processor:

DEFTAB

NAMTAB

ARGTAB

Definition table (DEFTAB)

- All the macro definitions in the program are stored in DEFTAB, which includes macro prototype and macro body statements.
- Comment lines from macro definition are not entered into DEFTAB because they will not be a part of macro expansion.
- References to the macro instruction parameters are converted to a positional notation for efficiency in substituting arguments.

Name table (NAMTAB)

- The macro names are entered into NAMTAB
- NAMTAB contains pointers to beginning and end of definition in DEFTAB.

Argument table (ARGTAB)

- The third data structure is an argument table (ARGTAB), which is used during expansion of macro invocations.
- When macro invocation statements are recognized, the arguments are stored in ARGTAB according to their position in argument list.
- As the macro is expanded, arguments from ARGTAB are substituted for the corresponding parameters in the macro body.
- Example: Consider the following source code

```
SUM MACRO &X,&Y
LDA &X
MOV B
LDA &Y
ADD B
```



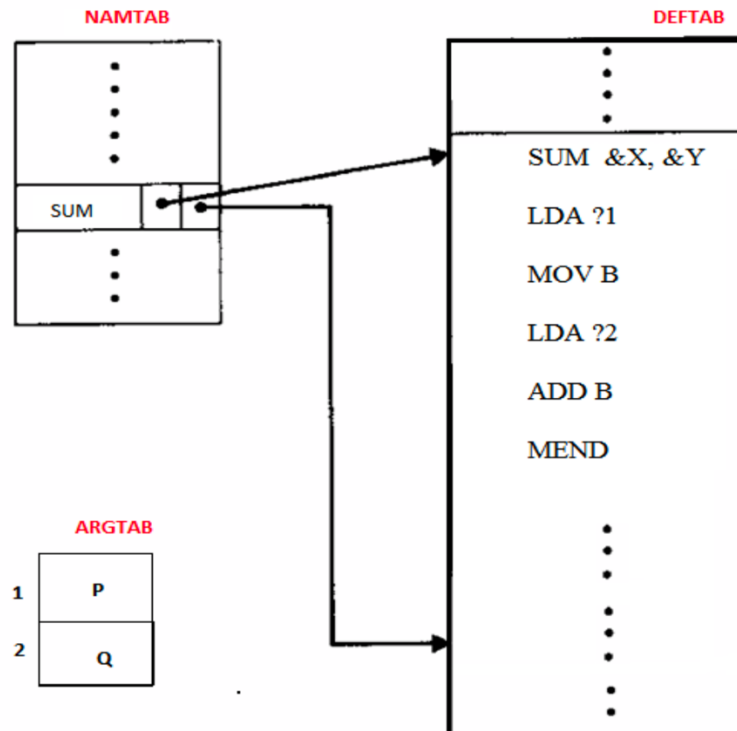
BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to University of Mumbai)
[Knowledge is Nectar]

Department of Computer Engineering

```
MEND
START
LDA 4500
ADD B
SUM P,Q
LDA 3000
.....
END
```

- When the macro definition for SUM is encountered, the macro name SUM along with its parameters X and Y are entered into DEFTAB. Then the statements in the body of macro is also entered into DEFTAB. The positional notation is used for the parameters. The parameter &X has been converted to ?1, &Y has been converted to ?2.
- The macro name SUM is entered into NAMTAB and the beginning and end pointers are also marked.
- On processing the input code, opcode in each statement is compared with the NAMTAB, to check whether it is a macro call. When the macro call SUM P,Q is recognized, the arguments P and Q will entered into ARGTAB. The macro is expanded by taking the statements from DEFTAB using the beginning and end pointers of NAMTAB.
- When the ?n notation is recognized in a line from DEFTAB, the corresponding argument is taken from ARGTAB

Figure shows the different data structures used



Input

; Define a macro to increment two values



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to University of Mumbai)
[Knowledge is Nectar]

Department of Computer Engineering

	<pre>INCR MACRO X, Y MOVER X, Y MEND ; Define a macro to decrement two values DECR MACRO X, Y MOVEM Y, X MEND ; Define a macro to print three values PRN MACRO X, Y, Z MOVER X, Y MOVEM Y, Z PRINT X, Y PRINT Z MEND START 100 READ N1 INCR N1, N2 DECR N1, N2 READ N2 INCR N1, N2 INCR N3, N4 DECR N3, N4 PRN A1, A2, Z9 STOP END</pre>
Implementation/ Code	<pre>from prettytable import PrettyTable class DefinitionTable: def __init__(self): self.index = None self.definition = None self.arg = [None, None] self.next = None class ArgumentListArray: def __init__(self): self.index = None self.arg = None self.next = None class NameTable: def __init__(self): self.index = None self.name = None self.dt_index = None self.next = None def find_arg_index(arg, al_head):</pre>



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to University of Mumbai)
[Knowledge is Nectar]

Department of Computer Engineering

```
temp = al_head
while temp is not None:
    if temp.arg == arg:
        return temp
    temp = temp.next
return None

def find_name(name, nt_head):
    temp = nt_head
    while temp is not None:
        if temp.name == name:
            return temp.dt_index
        temp = temp.next
    return None

def pass1(fp):
    global MDTC, MNTC
    MDTC = MNTC = 1
    dt_head = None
    nt_head = None
    al_head = None
    al_index = 1

    while True:
        line = fp.readline()
        if not line:
            break

        if "MACRO" in line:
            tokens = line.split()
            print(f"\nMACRO {tokens[0]} Detected...\n")

            if nt_head is None:
                nt_head = NameTable()
                nt_temp = nt_head
            else:
                nt_temp.next = NameTable()
                nt_temp = nt_temp.next

            nt_temp.index = MNTC
            MNTC += 1
            nt_temp.name = tokens[0]
            print(f"\n{tokens[0]} added into Name Table")

            for token in tokens[1:]:
                if token != "MACRO" and token != "\n":
                    if al_head is None:
```



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to University of Mumbai)
[Knowledge is Nectar]

Department of Computer Engineering

```
        al_head = ArgumentListArray()
        al_temp = al_head
    else:
        al_temp.next = ArgumentListArray()
        al_temp = al_temp.next

    al_temp.index = al_index
    al_index += 1
    al_temp.arg = token
    print(f"\nArgument {al_temp.arg} added into argument list
array")

    if dt_head is None:
        dt_head = DefinitionTable()
        dt_temp = dt_head
    else:
        dt_temp.next = DefinitionTable()
        dt_temp = dt_temp.next

    dt_temp.definition = nt_temp.name
    print(f"\nDefinition table entry created for {nt_temp.name}")
    nt_temp.dt_index = dt_temp

    while True:
        line = fp.readline()
        if line.strip() == "MEND":
            break

        tokens = line.split()
        is_arg = 0
        index = 0

        for token in tokens:
            if is_arg == 0:
                if dt_head is None:
                    dt_head = DefinitionTable()
                    dt_temp = dt_head
                else:
                    dt_temp.next = DefinitionTable()
                    dt_temp = dt_temp.next

                dt_temp.index = MDTC
                MDTC += 1
                dt_temp.definition = token
                print(f"\nEntry appended for {dt_temp.definition} at index
{dt_temp.index}")

                is_arg = 1
```




BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to University of Mumbai)
[Knowledge is Nectar]

Department of Computer Engineering

```
        else:
            if find_arg_index(token, al_head) is None:
                if al_head is None:
                    al_head = ArgumentListArray()
                    al_temp = al_head
                else:
                    al_temp.next = ArgumentListArray()
                    al_temp = al_temp.next

            al_temp.index = al_index
            al_index += 1
            al_temp.arg = token
            dt_temp.arg[index] = al_temp
        else:
            dt_temp.arg[index] = find_arg_index(token, al_head)
            index += 1

    # print("\nAll three tables are updated. Pass 1 Complete!\n")
    # Assuming nt_head, dt_head, and al_head are initialized in the main function
    print_name_table(nt_head)
    print_definition_table(dt_head)
    print_argument_list_array(al_head)

def pass2(fp):
    line = fp.readline()
    while line:
        print(line)
        temp = find_name(line, nt_head)
        if temp is not None:
            while temp.definition != "MEND":
                print("-", temp.definition, temp.arg[0], temp.arg[1])
                temp = temp.next
        line = fp.readline()

    print("\nOutput file updated with expanded code. Pass 2 Complete!\n")

def print_name_table(nt_head):
    table = PrettyTable(["Index", "Name", "Definition Table Index"])
    temp = nt_head
    while temp:
        table.add_row([temp.index, temp.name, temp.dt_index.index])
        temp = temp.next
    print("Name Table:")
    print(table)

def print_definition_table(dt_head):
```



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to University of Mumbai)
[Knowledge is Nectar]

Department of Computer Engineering

```
table = PrettyTable(["Index", "Definition", "Arguments", "Next"])
temp = dt_head
while temp:
    arg_list = [arg.arg for arg in temp.arg if arg]
    table.add_row([temp.index, temp.definition, arg_list, temp.next])
    temp = temp.next
print("\nDefinition Table:")
print(table)

def print_argument_list_array(al_head):
    table = PrettyTable(["Index", "Argument", "Next"])
    temp = al_head
    while temp:
        table.add_row([temp.index, temp.arg, temp.next])
        temp = temp.next
    print("\nArgument List Array:")
    print(table)

def main():
    global nt_head, al_head
    nt_head = None
    al_head = None

    try:
        with open("input.asm", "r") as fp:
            print("\nPass 1 in progress\n")
            pass1(fp)

        with open("input.asm", "r") as fp:
            print("\nPass 2 in progress\n")
            pass2(fp)

    except IOError:
        print("\nFailed to open the assembly file!")

if __name__ == "__main__":
    main()
```



**BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY**
(Empowered Autonomous Institute Affiliated to University of Mumbai)
[Knowledge is Nectar]

Department of Computer Engineering

Output

```
Pass 1 in progress

MACRO INCR Detected...

INCR added into Name Table
Argument X, added into argument list array
Argument Y added into argument list array
Definition table entry created for INCR
Entry appended for MOVER at index 1
MACRO DECR Detected...

DECR added into Name Table
Argument X, added into argument list array
Argument Y added into argument list array
Definition table entry created for DECR
Entry appended for MOVEM at index 2
MACRO PRN Detected...

PRN added into Name Table
Argument X, added into argument list array
Argument Y, added into argument list array
Argument Z added into argument list array
Definition table entry created for PRN
Entry appended for MOVER at index 3
Entry appended for MOVEM at index 4
Entry appended for PRINT at index 5
Entry appended for PRINT at index 6
Name Table
```



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to University of Mumbai)
[Knowledge is Nectar]

Department of Computer Engineering

Name Table:

Index	Name	Definition Table Index
1	INCR	None
2	DECR	None
3	PRN	None

Definition Table:

Index	Definition	Arguments	Next
None	INCR	[]	<__main__.DefinitionTable object at 0x10095baa0>
1	MOVER	['X,', 'Y']	<__main__.DefinitionTable object at 0x10098d520>
None	DECR	[]	<__main__.DefinitionTable object at 0x10098dcd0>
2	MOVEM	['Y,', 'X']	<__main__.DefinitionTable object at 0x10098dd60>
None	PRN	[]	<__main__.DefinitionTable object at 0x10098dee0>
3	MOVER	['X,', 'Y']	<__main__.DefinitionTable object at 0x10098e030>
4	MOVEM	['Y,', 'Z']	<__main__.DefinitionTable object at 0x10098e0f0>
5	PRINT	['X,', 'Y']	<__main__.DefinitionTable object at 0x10098e210>
6	PRINT	['Z']	None

Argument List Array:

Index	Argument	Next
1	X,	<__main__.ArgumentListArray object at 0x100959130>
2	Y	<__main__.ArgumentListArray object at 0x10098d3d0>
3	X,	<__main__.ArgumentListArray object at 0x10098de20>
4	Y	<__main__.ArgumentListArray object at 0x10098e060>
5	Y,	<__main__.ArgumentListArray object at 0x10098dd60>
6	X	<__main__.ArgumentListArray object at 0x10098e210>
7	X,	<__main__.ArgumentListArray object at 0x10098dc10>
8	Y,	<__main__.ArgumentListArray object at 0x10098e120>
9	Z	None



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to University of Mumbai)
[Knowledge is Nectar]

Department of Computer Engineering

```
Pass 2 in progress

; Define a macro to increment two values
INCR MACRO X, Y
    MOVER X, Y
MEND

; Define a macro to decrement two values
DECR MACRO X, Y
    MOVEM Y, X
MEND

; Define a macro to print three values
PRN MACRO X, Y, Z
    MOVER X, Y
    MOVEM Y, Z
    PRINT X, Y
    PRINT Z
MEND

START 100
READ N1
INCR N1, N2
DECR N1, N2
READ N2
INCR N1, N2
INCR N3, N4
DECR N3, N4
PRN A1, A2, Z9
STOP
END
```

Conclusion

In conclusion, I successfully implemented macros, enhancing code readability, efficiency, and maintainability in the assembly language program.



**BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY**
(Empowered Autonomous Institute Affiliated to University of Mumbai)
[Knowledge is Nectar]

Department of Computer Engineering

References

- [1] CS303 System Software
<http://www.icet.ac.in/Uploads/Downloads/Module%205.pdf>
- [2] ChatGPT (April 26, 2024) Two Pass Macro Processor
<https://chat.openai.com/share/cf1f7ca3-7c4c-4223-a4e2-3a0c5372af82>