Macro Language

Introduction

- Macro instructions are single line abbreviations for groups of instructions.
- For every occurrence of macro instruction, the micro processing assembler will substitute the entire block.
- Macro instructions are usually considered an extension of the basic assembler language and the macro processor is viewed as an extension of the basic assembler algorithm.

Macro Instruction

- Consider the program

	A	**1, DATA**	**Add contents of DATA to register 1**
	A	2, DATA	Add content of DATA to register 2
	A	3, DATA	Add contents of DATA to register 3
	…		
	A	1, DATA	Add contents of DATA to register 1
	A	2, DATA	Add content of DATA to register 2
	A	3, DATA	Add contents of DATA to register 3
	…		
DATA	DC	F’5’	
	…		

Cont…

- The macro instruction can be written as

Start of definition	**MACRO**
Macro Name	[ ]
Sequence of abbreviation	…..
End of definition	MEND

Cont…

- The program with macro instruction

	MACRO	
	INCR	
	A	1, DATA
	A	2, DATA
	A	3, DATA
	MEND	
	…	
	INCR	
	…	
	INCR	
	…	
DATA	DC	F’5’
	…	

Expanded Program

	A	**1, DATA**
	A	2, DATA
	A	3, DATA

	A	**1, DATA**
	A	2, DATA
	A	3, DATA

Macro Instruction Arguments

- Macro facility consists of providing for arguments, or parameters, in macro calls.
- A parameter is called a macro instruction argument or dummy argument.
- Dummy argument specified on the macro name line and distinguished by ampersand(&).

Cont…

- Consider the program

		…	
	A	1, DATA1	Add contents of DATA1 to register 1
	A	2, DATA1	Add content of DATA1 to register 2
	A	3, DATA1	Add contents of DATA1 to register 3
	…		
	A	1, DATA2	Add contents of DATA2 to register 1
	A	2, DATA2	Add content of DATA2 to register 2
	A	3, DATA2	Add contents of DATA2 to register 3
	…		
DATA1	DC	F’5’	
DATA2	DC	F’10’	
	…		

Cont…

- The program can be re-written as

	MACRO		
	INCR	&ARG	Macro INCR has one argument
	A	1, &ARG	
	A	2, &ARG	
	A	3, &ARG	
	MEND		
	…		
	INCR	DATA1	DATA1 is used as operand
	…		
	INCR	DATA2	DATA2 is used as operand
	…		
DATA1	DC	F’5’	
DATA2	DC	F’10’	
	…		

A	**1, DATA1**
A	2, DATA1
A	3, DATA1

A	**1, DATA2**
A	2, DATA2
A	3, DATA2

Cont…

Is it possible to supply more than one argument in a macro call?

Cont…

- Consider the program

	…	
LOOP1	A	1, DATA1
	A	2, DATA2
	A	3, DATA3
	…	
LOOP2	A	1, DATA3
	A	2, DATA2
	A	3, DATA1
	…	
DATA1	F‘5’	
DATA2	F‘10’	
DATA3	F‘15’	

Cont…

- The program can be written as

	…	
	MACRO	
&LAB	INCR	&ARG1, &ARG2, &ARG3
&LAB	A	1, ARG1
	A	2, ARG2
	A	3, ARG3
	MEND	
	…	
LOOP1	INCR	DATA1, DATA2, DATA3
	…	
LOOP2	INCR	DATA3, DATA2, DATA1
	…	
DATA1	F‘5’	
DATA2	F‘10’	
DATA3	F‘15’	

LOOP1	**A**	**1, DATA1**
	A	2, DATA2
	A	3, DATA3

LOOP2	**A**	**1, DATA3**
	A	2, DATA2
	A	3, DATA1

Cont…

- The program can also be written as

	…	
	MACRO	
	INCR	&ARG1, &ARG2, &ARG3, &LAB
&LAB	A	1, ARG1
	A	2, ARG2
	A	3, ARG3
	MEND	
	…	
	INCR	DATA1, DATA2, DATA3, LOOP1
	…	
	INCR	DATA3, DATA2, DATA1, LOOP2
	…	
DATA1	F‘5’	
DATA2	F‘10’	
DATA3	F‘15’	

Ways of Specifying Arguments

- There are following ways of specifying arguments to a macro call
  - Positional Argument
  - Keyword Argument
  - Default Specification Argument
  - Macros with mixed argument list
  - Other use of argument

Positional Arguments

- Arguments are matched with dummy arguments according to the order in which they appears
- A positional argument is written as &<parameter name>.

Cont…

- Consider the macro definition

MACRO	
INCR	&MEM_VAL, &INCR_VAL, &REG
MOVER	&REG, &MEM_VAL
ADD	&REG, &INCR_VAL
MOVEM	&REG, &MEM_VAL
MEND	

- The macro call

  INCR A, B, AREG

  Follow the rule of positional association.

Keyword Arguments

- It allows reference to dummy arguments
- For example, the macro definition

MACRO	
INCR_M	&MEM_VAL=, &INCR_VAL=, &REG=
MOVER	&REG, &MEM_VAL
ADD	&REG, &INCR_VAL
MOVEM	&REG, &MEM_VAL
MEND	

uses keyword arguments

- The macro call for this is
- INCR_M MEM_VAL=A, INCR_VAL = B, REG=AREA

Default Specification of Arguments

- Consider the macro definition

MACRO	
INCR_D	&MEM_VAL=, &INCR_VAL=, &REG=AREG
MOVER	&REG, &MEM_VAL
ADD	&REG, &INCR_VAL
MOVEM	&REG, &MEM_VAL
MEND	

- The macro call

  INCR_D MEM_VAL=A, INCR_VAL=B

  Use the default specification of an argument

Macro with Mixed Arguments

- A macro may be defined to use both positional and keyword parameters.
- For example, in the macro call

  SUMUP A, B, G=20, H=X

  A, B are positional parameters while G, H are keyword parameters

Other uses of Arguments

- The use of formal parameters are not restricted to use only in operand field.
- Formal parameters can also appear in the label and opcode of model statement.

	MACRO	
	CALC	&X, &Y, &OP = MULT, &LAB=
&LAB	MOVER	AREG, &X
	&OP	AREG, &Y
	MOVEM	AREG, &X
	MEND	

- The macro call, CALC A, B, LAB=LOOP

Conditional Macro Expansion

- Two important macro pseudo-ops, AIF and AGO, permit conditional reordering of the sequence of macro expansion.

Cont…

- Consider the program

		…	
LOOP1	A	1, DATA1	Add contents of DATA1 to register 1
	A	2, DATA2	Add content of DATA2 to register 2
	A	3, DATA3	Add contents of DATA3 to register 3
	…		
LOOP2	A	1, DATA3	Add contents of DATA3 to register 1
	A	2, DATA2	Add content of DATA2 to register 2
	…		
LOOP3	A	1, DATA1	Add contents of DATA1 to register 1
	…		
DATA1	DC	F’5’	
DATA2	DC	F’10’	
DATA3	DC	F’15’	
	…		

	MACRO		
&ARGO	VARY	&COUNT,&ARG1,&ARG2,&ARG3	
&ARGO	A	1,&ARG1	
	AIF	(&COUNT EQ 1).FINI	Test if &COUNT =1
	A	2, &ARG2	
	AIF	(&COUNT EQ 2).FINI	Test if &COUNT =2
.FINI	MEND		
	…		
LOOP1	VARY	3, DATA1, DATA2, DATA3	
	…		
LOOP2	VARY	2, DATA3, DATA2	
	…		
LOOP3	VARY	1, DATA1	
	…		
DATA1	DC	F’5’	
DATA2	DC	F’10’	
DATA3	DC	F’15’	
	…		

Cont…

- Label starting with a period (.) such as .FINI, are macro labels and do not appear in the output of the macro processor.
- The statement AIF (&COUNT EQ 1).FINI directs the macro processor to skip to the statement labeled .FINI if the parameter corresponding to &COUNT is 1; otherwise, the macro processor is to continue with the statement.

Cont…

- AIF is a conditional branch pseudo-op; it performs an arithmetic test and branches only if the tested condition is true.
- The AGO is an unconditional branch pseudo-op or ‘go-to’ statement. It specifies a label appearing on some other statement in the macro instruction definition.
- AIF and AGO are directives to the macro processor and do not appear in macro expansion.

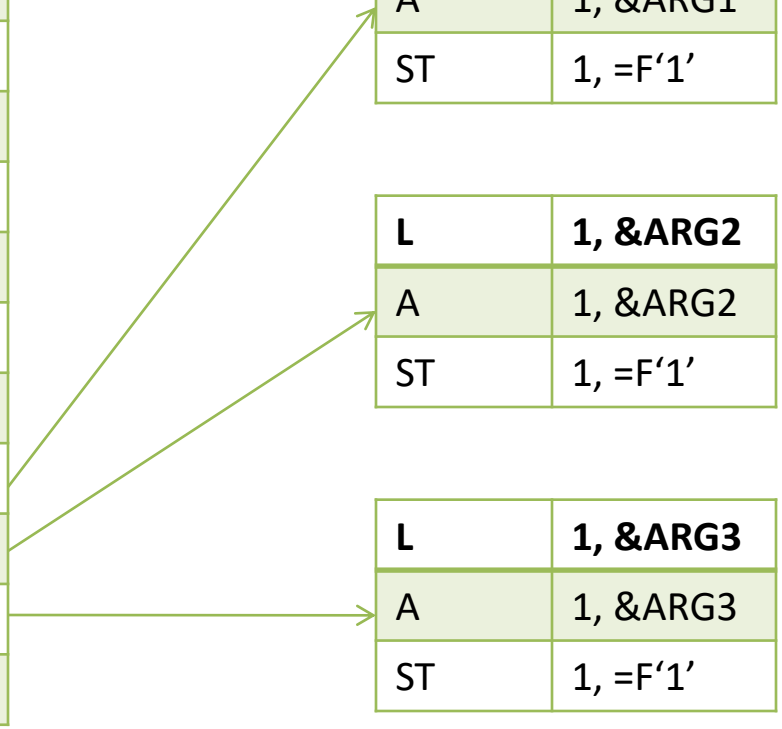
Macro calls within Macros

- Since macro calls are abbreviations of instruction sequence, it seems reasonable that such abbreviations should be available within other macro definition.

Cont…

- Consider the example

MACRO	
ADD1	&ARG
L	1, &ARG
A	1, &ARG
ST	1, =F‘1’
MEND	
MACRO	
ADDS	&ARG1, &ARG2, &ARG3
ADD1	&ARG1
ADD1	&ARG2
ADD1	&ARG3
MEND	



L	**1, &ARG1**
A	1, &ARG1
ST	1, =F‘1’

L	**1, &ARG2**
A	1, &ARG2
ST	1, =F‘1’

L	**1, &ARG3**
A	1, &ARG3
ST	1, =F‘1’

Implementation

- There are four basic tasks that any macro instruction processor must perform
  - Recognize macro definitions
  - Save the definition
  - Recognize calls
  - Expand calls

A Two-Pass Algorithm

- For designing a macro processor, assume that it is independent of assembler and its output is fed into the assembler.
- A macro processor scans and processes lines of text.
- A macro processor algorithm will make two systematic scans, or passes, over input text.
  - First for handling the macro definitions
  - Second for handling the macro calls

Cont…

- The first pass examine every operation code and save all macro definitions in a Macro Definition Table (MDT).
- The first pass also prepare a Macro Name Table (MNT).
- The second pass examine every operation mnemonic and replace each macro name with the appropriate text from the definition.

Specification of Databases

- Pass-I Databases
  - The input macro source deck
  - The output macro source deck
  - The Macro Definition Table (MDT)
  - The Macro Name Table (MNT)
  - The Macro Definition Table Counter (MDTC)
  - The Macro Name Table Counter (MNTC)
  - The Argument List Array (ALA)

Cont…

- Pass-II Databases
  - The input macro source deck
  - The output expanded source deck
  - The Macro Definition Table(MDT)
  - The Macro Name Table(MNT)
  - The Macro Definition Table Pointer (MDTP)
  - The Argument List Array (ALA)

Database Formats

Macro Definition Table

- The Macro Definition Table (MDT) is a table of text line.
- Every line of each macro definition, except MACRO line, is stored in the MDT.
- The MEND is kept to indicate the end of the definition

Cont…

- Consider the example

	…	
	MACRO	
&LAB	INCR	&ARG1, &ARG2, &ARG3
&LAB	A	1, ARG1
	A	2, ARG2
	A	3, ARG3
	MEND	
	…	
LOOP1	INCR	DATA1, DATA2, DATA3
	…	
LOOP2	INCR	DATA3, DATA2, DATA1
	…	
DATA1	F‘5’	
DATA2	F‘10’	
DATA3	F‘15’	

	Macro Definition Table		
Index	Instructions		
…		…	
15	&LAB	INCR	&ARG1, &ARG2, &ARG3
16	0#	A	1, #1
17		A	2, #2
18		A	3, #3
19		MEND	
…		…	

Argument List Array(ALA)

- Consider the example

	…	
	MACRO	
&LAB	INCR	&ARG1, &ARG2, &ARG3
&LAB	A	1, ARG1
	A	2, ARG2
	A	3, ARG3
	MEND	
	…	
LOOP1	INCR	DATA1, DATA2, DATA3
	…	
LOOP2	INCR	DATA3, DATA2, DATA1
	…	
DATA1	F‘5’	
DATA2	F‘10’	
DATA3	F‘15’	

Argument List Array	
Index	**Argument**
0	LOOP1
1	DATA1
2	DATA2
3	DATA3

Macro Name Table

- The Macro Name Table (MNT) serves a function very similar to that assembler's Machine-Op Table.
- Each MNT entry consists of a character string and a pointer.

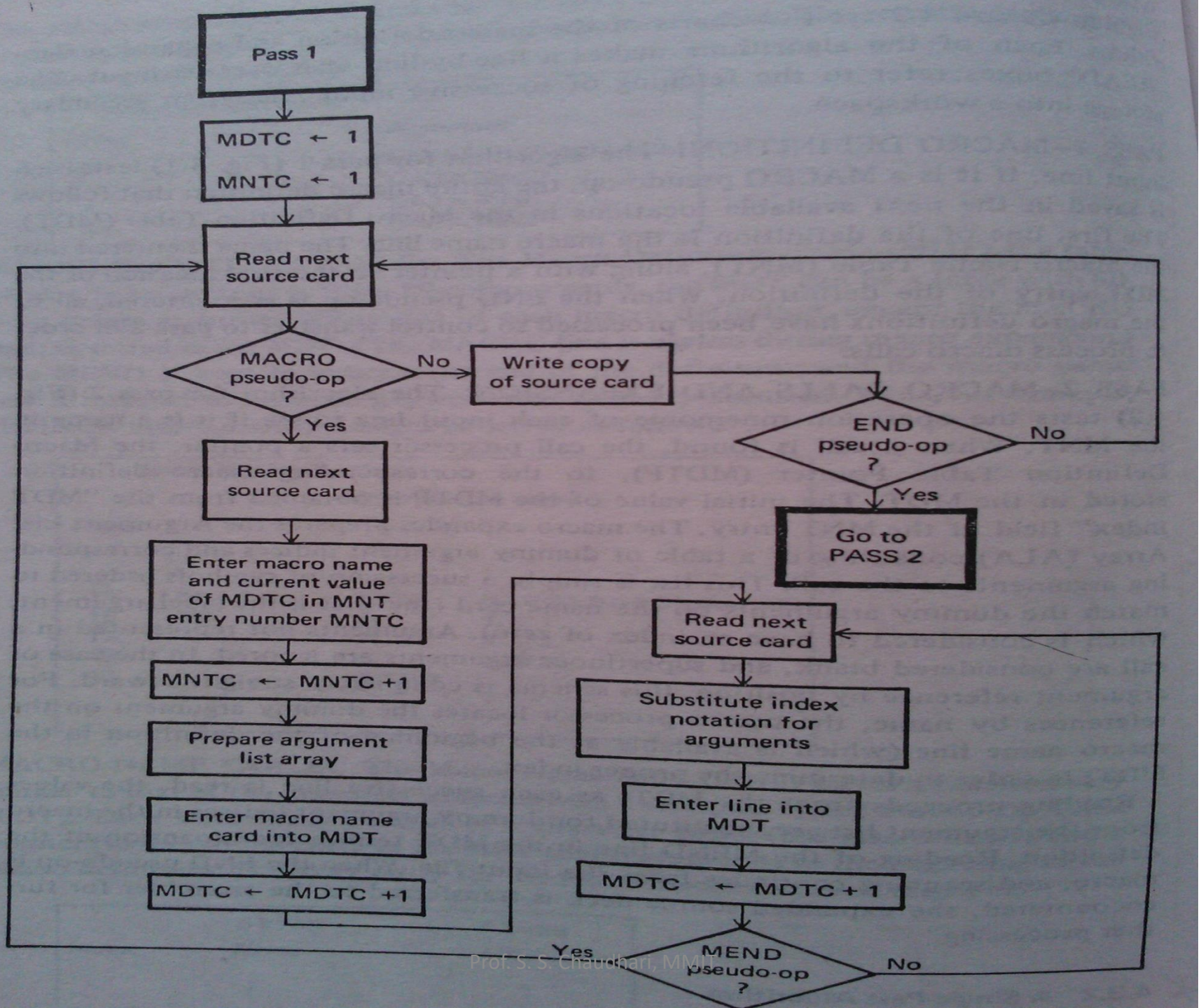
Cont…

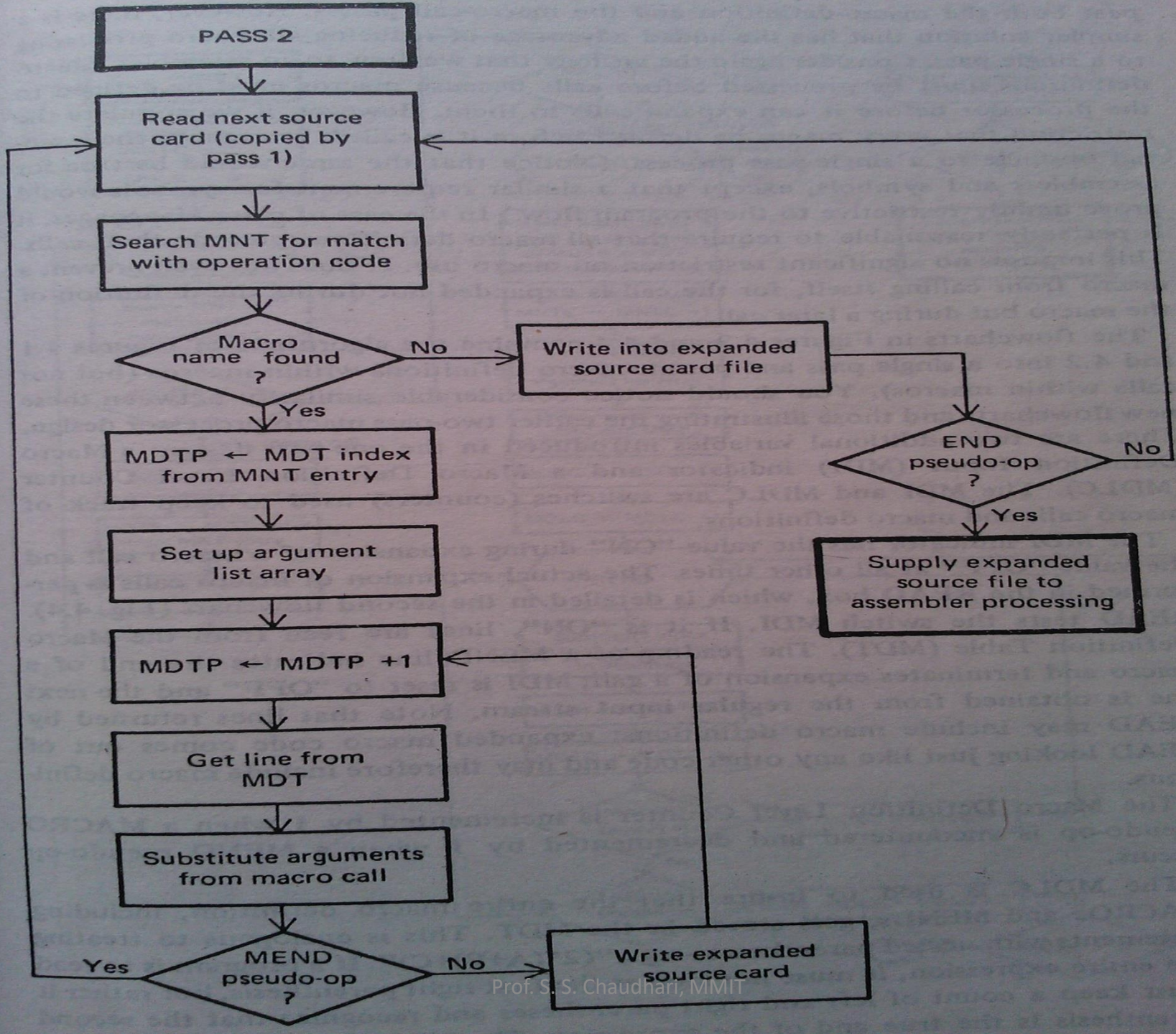
- Consider the example

	…	
	MACRO	
&LAB	INCR	&ARG1, &ARG2, &ARG3
&LAB	A	1, ARG1
	A	2, ARG2
	A	3, ARG3
	MEND	
	…	
LOOP1	INCR	DATA1, DATA2, DATA3
	…	
LOOP2	INCR	DATA3, DATA2, DATA1
	…	
DATA1	F‘5’	
DATA2	F‘10’	
DATA3	F‘15’	

Macro Name Table		
Index	**Name**	**MDT Index**
…	…	…
3	INCR	15
…	…	….

Algorithms





Macros and Macro Processor

Macro Instruction

- Macro instructions are single line abbreviations for groups of instructions.
- For every occurrence of macro instruction, the micro processing assembler will substitute the entire block.
- Macros are used to provide a program generation facility through macro expansion
- A macro consists of a name, a set of formal parameters and a body of code.

Macro Definition

- A macro definition is enclosed between a macro header statement (MACRO) and a macro end statement(MEND).
- Macro definition are typically located at the start of a program.
- A macro definition consists of
  - A macro prototype statement
  - One or more model statement
  - Macro preprocessor statement

Cont…

- The macro prototype statement declare the name of a macro and the parameters used.
- A model statement is a statement from which an assembly language statement may be generated.
- A preprocessor statement is used to perform auxiliary functions during macro expansion.

Start of definition	**MACRO**
Macro Name	[ ]
Sequence of abbreviation	…..
End of definition	MEND

Syntax for writing the macro

Cont…

- The macro prototype has the following syntax

  <macro name>[<format parameter spec>[…]]

- For example,

MACRO	
INCR	&MEM_VAL, &INCR_VAL, &REG
MOVER	&REG, &MEM_VAL
ADD	&REG, &INCR_VAL
MOVEM	&REG, &MEM_VAL
MEND	

Macro Calls

- A macro is called by writing the macro name in the mnemonic field.
- The macro call has the syntax

  <macro name> [<actual parameter spec>[,…]]

  where actual parameter typically resembles an operand specification

Example

- Consider the assembly language program

	A	**1, DATA**	**Add contents of DATA to register 1**
	A	2, DATA	Add content of DATA to register 2
	A	3, DATA	Add contents of DATA to register 3
	…		
	A	1, DATA	Add contents of DATA to register 1
	A	2, DATA	Add content of DATA to register 2
	A	3, DATA	Add contents of DATA to register 3
	…		
DATA	DC	F’5’	
	…		

Cont…

- The program can be written using macros as

	MACRO	
	INCR	
	A	1, DATA
	A	2, DATA
	A	3, DATA
	MEND	
	…	
	INCR	
	…	
	INCR	
	…	
DATA	DC	F’5’
	…	

Expanded Program

	A	**1, DATA**
	A	2, DATA
	A	3, DATA

	A	**1, DATA**
	A	2, DATA
	A	3, DATA

Macro Expansion

- A macro call leads to macro expansion.
- During macro expansion, the macro call statement is replaced by a sequence of assembly statements.

Macro Instruction Arguments

- Macro facility consists of providing for arguments, or parameters, in macro calls.
- A parameter is called a macro instruction argument or dummy argument.
- Dummy argument specified on the macro name line and distinguished by ampersand(&).

Cont…

- Consider the program

		…	
	A	1, DATA1	Add contents of DATA1 to register 1
	A	2, DATA1	Add content of DATA1 to register 2
	A	3, DATA1	Add contents of DATA1 to register 3
	…		
	A	1, DATA2	Add contents of DATA2 to register 1
	A	2, DATA2	Add content of DATA2 to register 2
	A	3, DATA2	Add contents of DATA2 to register 3
	…		
DATA1	DC	F’5’	
DATA2	DC	F’10’	
	…		

Cont…

- The program can be written as

	MACRO		
	INCR	&ARG	Macro INCR has one argument
	A	1, &ARG	
	A	2, &ARG	
	A	3, &ARG	
	MEND		
	…		
	INCR	DATA1	DATA1 is used as operand
	…		
	INCR	DATA2	DATA2 is used as operand
	…		
DATA1	DC	F'5'	
DATA2	DC	F'10'	
	…		

A	**1, DATA1**
A	2, DATA1
A	3, DATA1

A	**1, DATA2**
A	2, DATA2
A	3, DATA2

Cont…

Is it possible to supply more than one argument in a macro call?

Cont…

- Consider the program

	…	
LOOP1	A	1, DATA1
	A	2, DATA2
	A	3, DATA3
	…	
LOOP2	A	1, DATA3
	A	2, DATA2
	A	3, DATA1
	…	
DATA1	F‘5’	
DATA2	F‘10’	
DATA3	F‘15’	

Cont…

- The program can be written as

	…	
	MACRO	
&LAB	INCR	&ARG1, &ARG2, &ARG3
&LAB	A	1, &ARG1
	A	2&ARG2
	A	3, &ARG3
	MEND	
	…	
LOOP1	INCR	DATA1, DATA2, DATA3
	…	
LOOP2	INCR	DATA3, DATA2, DATA1
	…	
DATA1	F‘5’	
DATA2	F‘10’	
DATA3	F‘15’	

LOOP1	**A**	**1, DATA1**
	A	2, DATA2
	A	3, DATA3

LOOP2	**A**	**1, DATA3**
	A	2, DATA2
	A	3, DATA1

Cont…

- The program can also be written as

	…	
	MACRO	
	INCR	&ARG1, &ARG2, &ARG3, &LAB
&LAB	A	1, DATA1
	A	2, DATA2
	A	3, DATA3
	MEND	
	…	
	INCR	DATA1, DATA2, DATA3, LOOP1
	…	
	INCR	DATA3, DATA2, DATA1, LOOP2
	…	
DATA1	F‘5’	
DATA2	F‘10’	
DATA3	F‘15’	

Ways of Specifying Arguments

- There are following ways of specifying arguments to a macro call
  - Positional Argument
  - Keyword Argument
  - Default Specification Argument
  - Macros with mixed argument list
  - Other use of argument

Positional Arguments

- Arguments are matched with dummy arguments according to the order in which they appears
- A positional parameter is written as &<parameter name>.

Cont…

- Consider the macro definition

MACRO	
INCR	&MEM_VAL, &INCR_VAL, &REG
MOVER	&REG, &MEM_VAL
ADD	&REG, &INCR_VAL
MOVEM	&REG, &MEM_VAL
MEND	

- The macro call

  INCR A, B, AREG

  Follow the rule of positional association.

Keyword Arguments

- It allows reference to dummy arguments
- For example, the macro definition

MACRO	
INCR_M	&MEM_VAL=, &INCR_VAL=, &REG=
MOVER	&REG, &MEM_VAL
ADD	&REG, &INCR_VAL
MOVEM	&REG, &MEM_VAL
MEND	

uses keyword arguments

- The macro call for this is
- INCR_M MEM_VAL=A, INCR_VAL = B, REG=AREA

Default Specification of Arguments

- Consider the macro definition

MACRO	
INCR_D	&MEM_VAL=, &INCR_VAL=, &REG=AREG
MOVER	&REG, &MEM_VAL
ADD	&REG, &INCR_VAL
MOVEM	&REG, &MEM_VAL
MEND	

- The macro call

  INCR_D MEM_VAL=A, INCR_VAL=B

  Use the default specification of an argument

Macro with Mixed Arguments

- A macro may be defined to use both positional and keyword parameters.
- For example, in the macro call

  SUMUP A, B, G=20, H=X

  A, B are positional parameters while G, H are keyword parameters

Other uses of Arguments

- The use of formal parameters are not restricted to use only in operand field.
- Formal parameters can also appear in the label and opcode of model statement.

	MACRO	
	CALC	&X, &Y, &OP = MULT, &LAB=
&LAB	MOVER	AREG, &X
	&OP	AREG, &Y
	MOVEM	AREG, &X
	MEND	

- The macro call, CALC A, B, LAB=LOOP

Nested Macro Calls

- A macro statement in a macro may constitute a call on another macro. Such calls are known as nested macro calls.
- Expansion of nested macro calls follows the last-in-first-out(LIFO) rule.

Cont…

MACRO	
INCR_D	&MEM_VAL=, &INCR_VAL=, &REG=AREG
MOVER	&REG, &MEM_VAL
ADD	&REG, &INCR_VAL
MOVEM	&REG, &MEM_VAL
MEND	

MACRO	
COMPUTE	&FIRST, &SECOND
MOVEM	BREG, TMP
INCR_D	&FIRST, &SECOND, REG=BREG
MOVER	BREG, TMP
MEND	

Cont…

- An AIF statement has the syntax

  AIF (<expression>)<sequencing symbol>

  Where expression is a relational expression involving ordinary strings, formal parameters and attributes.
- If the relational expression evaluates to true, expansion time control is transferred to the statement containing sequencing symbol in its label field

Cont…

- An AGO statement has the syntax

  AGO<sequencing symbol>
- And unconditional transfers expansion time control to the statement containing <sequencing symbol> in its label field.
- An ANOP statement is written as

  <sequencing symbol> ANOP
- And simply has the effect of defining the sequencing symbol.

Expansion Time Variable

- Expansion time variables(EV) are variables which can only be used during the expansion of macro calls.
- A local EV is created for use only during a particular macro call.
- A global EV exists across all macro calls situated in a program

Cont…

- Local and global EV's are created through declaration statement

  LCL <EV spec>[,<EV spec>…]

  GBL <EV spec>[,<EV spec>…]

- <EV spec> has the syntax &<EV name>
- Values of EV's can be manipulated through the preprocessor statement SET.

  <EV spec> SET <SET-expression>

- The SET statement assigns the value of <SET-expression> to the EV

Example

	MACRO	
	CONSTANTS	
	LCL	&A
&A	SET	1
	DB	&A
&A	SET	&A+1
	DB	&A
	MEND	

Attributes of Formal Parameter

- An attributes is written using syntax

  <attribute name>’<formal parameter spec>
- It represents the information about the value of the formal parameter, i.e., the type, length and size.

	MACRO	
	DCL_CONST	&A
	AIF	(L’ &A EQ 1) .NEXT
	…	
.NEXT	…	
	…	
	MEND	

Conditional Expansion

- Conditional expansion helps in generating assembly code specifically suited to the parameter in a macro call.
- The AIF and AGO statements are used for this purpose.

	MACRO	
	EVAL	&X, &Y, &Z
	AIF	(&Y EQ &X) .ONLY
	MOVER	AREG, &X
	SUB	AREG, &Y
	ADD	AREG, &Z
	AGO	.OVER
.ONLY	MOVER	AREG, &Z
.OVER	MEND	

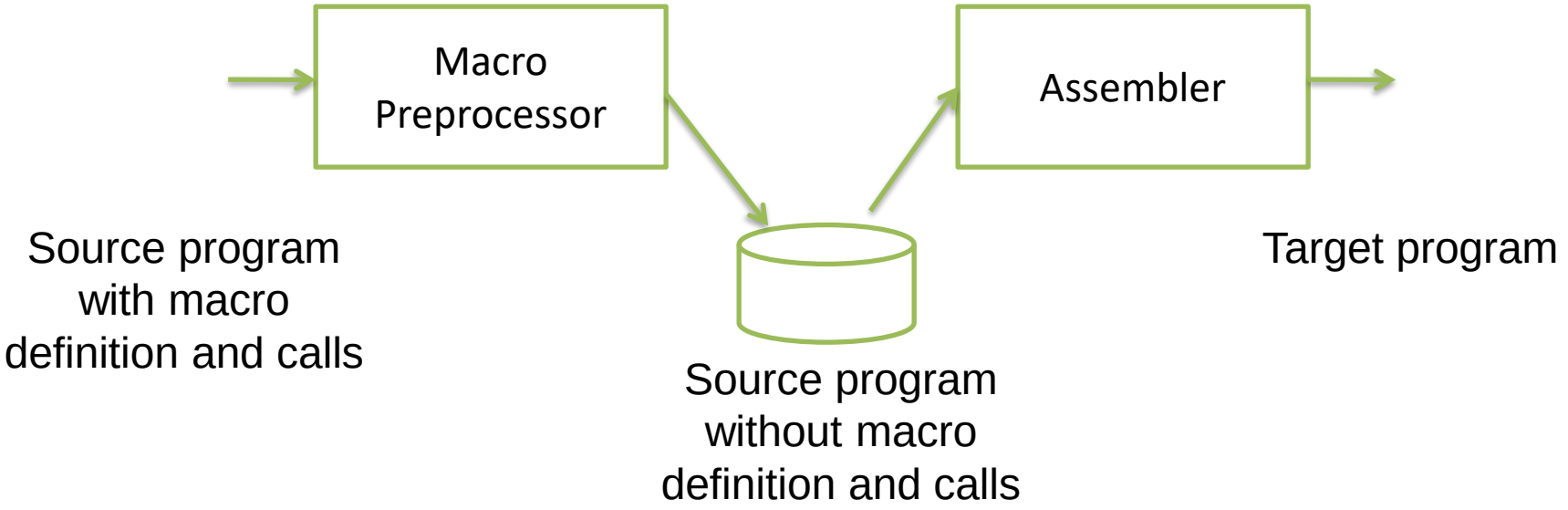
Expansion Time Loop

- It is often necessary to generate many similar statements during the expansion of a macro.
- This can be achieved by writing similar model statement in the macro.

MACRO	
CLEAR	&A
MOVER	AREG, =‘0’
MOVEM	AREG, &A
MOVEM	AREG, &A+1
MOVEM	AREG, &A+2
MEND	

Design of Macro Preprocessor

- The macro preprocessor accepts an assembly program containing definitions and calls and translate it into an assembly program which does not contain any macro definition or calls.



Design Overview

- We begin design by listing all tasks involved in macro expansion
  - Identify macro calls in the program
  - Determine the valued of formal parameters
  - Maintain the values of expansion time variable declared in a macro.
  - Organize expansion time control flow.
  - Determine the values of sequencing symbols.
  - Perform expansion of a model statement.

Cont…

- Design specification of each task is achieved by
  - Identify the information necessary to perform a task.
  - Design a suitable data structure to record the information.
  - Determine the processing necessary to obtain the information.
  - Determine the processing necessary to perform the task.

Cont…

- Identify macro calls
  - A table called the macro name table (MNT) is designed to hold the name of all macros defined in a program.
  - A macro name is entered in the table when a macro definition is processed.
- Determine values of formal parameters
  - A table called the actual parameter table(APT) is designed to hold the values of formal parameters during the expansion of macro call.
  - Each entry in the table is a pair

    (<formal parameter name>, <value>)

Cont…

- Maintain expansion time variable
  - An expansion time variable table (EVT) is maintained for this purpose.
  - The table contains pair of the form

    (<EV name>, <value>)
- Organize expansion time control flow
  - The flow of control during macro expansion determines when a model statement is to be visited for expansion.

Cont…

- Determine values of sequencing symbols
  - A sequencing symbol table is maintained to hold this information.
  - The table contains pair of the form

    (<sequencing symbol name>, <MDT entry #>)
- Perform expansion of a model statement
  - Expansion of a model statement is achieved by performing a lexical substitution for parameters and EV's used in the model.