**CSCI 360 Macros**

A macro is an extension to the basic ASSEMBLER language. They provide a means for generating a commonly used sequence of assembler instructions/statements. The sequence of instructions/statements will be coded ONE time within the macro definition. Whenever the sequence is needed within a program, the macro will be "called".

A macro definition precedes all CSECTs and DSECTs. It consists of four parts:

1. The macro instruction MACRO starting in column 10
2. The prototype statement (this line specifies the macro name and the arguments that it takes)
3. The macro body
4. The macro instruction MEND starting in column 10

MACRO signals the beginning of a macro definition.

**Prototype format:**

label macro-name arguments (0 to 200 possible)

Column: 1 10 16(usually)

The label in the prototype is optional. It can be used to put a label at the beginning of one of the lines of the loop body.

The macro body contains the macro instructions to be executed and the assembly instructions to be copied into the code.

MEND signals the end of a macro definition.

For example:

MACRO

EXMPL1

LA 1,PARMLIST

L 15,=V(BUILD)

BALR 14,15

MEND

To call the EXMPL1 macro:

EXMPL1

In the assembly, you'll see:

+ LA 1,PARMLIST

+ L 15,=V(BUILD)

+ BALR 14,15

The + in column 1 indicates that the lines of code were in a macro.

**Variable Symbols**

**Variable symbols** are symbols that can be assigned values by either the programmer or the assembler. There are three types:

1. Symbolic Parameters
2. System Variables
3. SET Variables

Variable symbol names are:

1. Two to eight characters in length
2. The first character is ALWAYS an ampersand (&)
3. The second character is ALWAYS a letter
4. The rest of the characters are either letters or digits

**Symbolic Parameters** are used in a macro definition and are assigned a value by the programmer. When the macro is called, these parameters are replaced by the values that are assigned to them. There are two types of symbolic parameters:

1. Positional Parameters
2. Keyword Parameters

**Positional parameters** are symbolic parameters that must be specified in a specific order every time the macro is called. The parameter will be replaced within the macro body by the value specified when the macro is called.

MACRO

&LABEL EXMPL1 &SUBRTN,&PARMS

&LABEL LA 1,&PARMS

L 15,=V(&SUBRTN)

BALR 14,15

MEND

The new call of the EXMPL1 macro:

CALL1 EXMPL1 BUILD,PARM1

In the assembly:

+CALL1 LA 1,PARM1

+ L 15,=V(BUILD)

+ BALR 14,15

**Keyword parameters** are symbolic parameters that can be specified in any order when the macro is called. The parameter will be replaced within the macro body by the value specified when the macro is called. These parameters can be given a default value. If no default value is specified and if the parameter is not given a value when the macro is called, then the parameter will be replaced by a null string.

Each keyword parameter will have an equal sign (=) as the last character of the parameter name.

MACRO

&LABEL EXMPL1 &SUBRTN=BUILD,&PARMS=

&LABEL LA 1,&PARMS

L 15,=V(&SUBRTN)

BALR 14,15

MEND

The new call of the EXMPL1 macro:

EXMPL1 PARMS=PARM2,SUBRTN=PRINT

In the assembly:

+ LA 1,PARM2

+ L 15,=V(PRINT)

+ BALR 14,15

EXMPL1 PARMS=PARM3

In the assembly:

+ LA 1,PARM3

+ L 15,=V(BUILD)

+ BALR 14,15

EXMPL1

In the assembly:

+ LA 1,

+ L 15,=V(BUILD)

+ BALR 14,15

If a combination of positional and keyword parameters is used, all of the positional parameters must be coded BEFORE the keyword parameters.

Prototype: &LABEL EXMPL2 &D,&E,&A=,&B=,&C=20

Call 1: CALL1 EXMPL2 VAL3,VAL4,B=VAL1,A=(R5,R7),C=

D will get VAL3, E will get VAL4, A will get (R5,R7),

B will get VAL1 and C will get the null string

Call 2: CALL2 EXMPL2 VAL3,,C=10,A=25

D will get VAL3, E will get the null string, A will get 25,

B will get the null string and C will get 10

**System variables** are variables that are assigned values by the assembler. They are defined by the system.

The most common system variables are: **&SYSNDX, &SYSLIST, &SYSDATE, &SYSTIME, &SYSPARM, &SYSECT**

**&SYSNDX**

* Generates a unique 4 digit number each time the macro is called
* The value is usually concatenated to a symbol of 4 characters
* The first value is 0001, with each subsequent call the value will be incremented by one

MACRO

&LABEL ADD &NUM1,&NUM2

&LABEL ST 5,SAVE

L 5,&NUM1

A 5,&NUM2

ST 5,&NUM1

B NEXT

SAVE DC F'-1'

NEXT L 5,SAVE

MEND

In a program:

ADD FLD1,FLD2

+ ST 5,SAVE

+ L 5,FLD1

+ A 5,FLD2

+ ST 5,FLD1

+ B NEXT

+SAVE DC F'-1'

+NEXT L 5,SAVE

...

...

ADD FLD3,FLD4

+ ST 5,SAVE

+ L 5,FLD3

+ A 5,FLD4

+ ST 5,FLD3

+ B NEXT

+SAVE DC F'-1'

+NEXT L 5,SAVE

The labels SAVE and NEXT are duplicated when the macro is called for a second time, an assembler no-no. &SYSNDX can be used to solve this problem.

We're going to concatenate the &SYSNDX value onto the end of each of the labels.

MACRO

&LABEL ADD &NUM1,&NUM2

&LABEL ST 5,SAVE&SYSNDX

L 5,&NUM1

A 5,&NUM2

ST 5,&NUM1

B NEXT&SYSNDX

SAVE&SYSNDX DC F'-1'

NEXT&SYSNDX L 5,SAVE&SYSNDX

MEND

In a program:

ADD FLD1,FLD2

+ ST 5,SAVE0001

+ L 5,FLD1

+ A 5,FLD2

+ ST 5,FLD1

+ B NEXT0001

+SAVE0001 DC F'-1'

+NEXT0001 L 5,SAVE0001

...

...

ADD FLD3,FLD4

+ ST 5,SAVE0002

+ L 5,FLD3

+ A 5,FLD4

+ ST 5,FLD3

+ B NEXT0002

+SAVE0002 DC F'-1'

+NEXT0002 L 5,SAVE0002

**&SYSLIST**

* Used to refer to a positional parameter or an entry in a positional parameter sublist

Format 1: &SYSLIST(n) -- refers to the nth positional parameter

Format 2: &SYSLIST(n,k) -- refers to the nth positional parameter's

kth member

CALL3 EXMPL3 ONE,TWO,(R5,R6,R7,R8),,FIVE,(R1)

(R5,R6,R7,R8) is a positional parameter sublist with 4 members

(R1) is a positional parameter sublist with 1 member

&SYSLIST(2) --> TWO

&SYSLIST(3,3) --> R7

&SYSLIST(4) --> null value

&SYSLIST(6,1) --> R1

Keyword parameters may also have sublists, but &SYSLIST CANNOT be used to reference the values. The name of the keyword parameter will be used.

MACRO

SUBLISTS &P1,&KEY=(0,1,3)

&P1(1) DC F'&KEY(1)'

&SYSLIST(1,2) EQU &P1(2)

MEND

&KEY --> (0,1,3)

&KEY(1) --> 0

&KEY(2) --> 1

&KEY(3) --> 3

&KEY(4) --> NULL

In a program:

CALL4 SUBLISTS (HERE,THERE),KEY=(5,6,7)

+HERE DC F'5'

+THERE EQU THERE

**&SYSDATE** is used to obtain the date that the source code was assembled. It is in the format MM/DD/YY

**&SYSTIME** is used to obtain the time that the source code was assembled. It is in the format hh.mm

**&SYSECT** is used to obtain the name of CSECT that the macro was called in.

**&SYSPARM** is used to pass a character string from the JCL.

**SET variables** are parameters that are assigned an initial value and can be altered during the macro call. There are three types:

1. Arithmetic
2. Binary
3. Character

Each SET variable can be either ***local*** or ***global***.

A local SET variable is assigned an initial value every time the macro is called. It is known only to the macro it is created in.

A global SET variable is initialized the first time the macro is called and retains a value from one call to another. They can be referenced within other macros also, but they must be declared within each macro.

All SET variables must be declared directly after the prototype statement. All global variables must be declared before local variables.

**Arithmetic SET variables** are variables whose value can be altered by an arithmetic expression.

**To declare a LOCAL variable:** LCLA &var\_name1,&var\_name2...

- These are initialized to 0 every time the macro is called

**To declare a GLOBAL variable:** GBLA &var\_name1,&var\_name2...

- These are initialized to 0 the first time the macro is called

**To alter the value:** &var\_name SETA arithmetic\_expression

LCLA &CNTR declares an arithmetic variable called &CNTR

&CNTR SETA 1 changes the counter value to 1

&CNTR SETA &CNTR+1 increments the counter value by 1

&CNTR SETA &CNTR\*5 multiply counter by 5

**Binary SET variables** are variables whose value can be either 0 or 1. They are typically used as flag variables with 0 being FALSE and 1 being TRUE.

**To declare a LOCAL variable:** LCLB &var\_name1,&var\_name2...

- These are initialized to 0 every time the macro is called

**To declare a GLOBAL variable:** GBLB &var\_name1,&var\_name2...

- These are initialized to 0 the first time the macro is called

**To alter the value:** &var\_name SETB (condition)

- (condition) will resolve to either a 0 (false) or 1 (true)

because of this, 0 or 1 can be hard-coded in place of the

(condition)

LCLB &FLAG

&FLAG SETB ('&PARMS' EQ '')

**Character SET variables** are variables whose value can be a text string of up to 255 characters.

**To declare a LOCAL variable:** LCLC &var\_name1,&var\_name2...

- These are initialized to the null string every time the macro

is called

**To declare a GLOBAL variable:** GBLC &var\_name1,&var\_name2...

- These are initialized to the null string the first time the

macro is called

**To alter the value:** &var\_name SETC string

- string can be:

1. A text string enclosed in single quotes 'hello'

2. A text string and character set symbol 'ABC&MSG' or '&MSG.ABC'

3. A combination of two character set symbols '&MSG1&MSG2'

4. A substring of a character set symbol '&MSG'(start,length)

- start is the first character of substring

- length is the length of the substring

LCLC &MSG

&MSG SETC 'Hello'

&MSG SETC '&MSG. world' changes &MSG to 'Hello world'

NOTE: The period is needed to signal the end of the set symbol name

LCLC &NDX

&NDX SETC '&SYSNDX' now &NDX can be used in place of &SYSNDX

LCLC &TEXT1,&TEXT2

&TEXT1 SETC 'BOBCAT'

&TEXT2 SETC '&TEXT1'(1,3) &TEXT2 set to 'BOB'

There are two types of comments that can be included in a macro definition.

A comment starting with a \* in column 1 will be reproduced in the assembly.

A comment starting with a .\* will NOT be reproduced in the assembly.

MACRO

&LABEL EXMPL1 &SUBRTN=BUILD,&PARMS=

.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.\*

.\* Definition for a macro to call an external subroutine

.\*

.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* The following lines will call an external subroutine

\*

&LABEL LA 1,&PARMS

L 15,=V(&SUBRTN)

BALR 14,15

MEND

The new call of the EXMPL1 macro:

EXMPL1 SUBRTN=PRINT,PARMS=PARM2

In the assembly:

+\*

+\* The following lines will call an external subroutine

+\*

+ LA 1,PARM4

+ L 15,=V(PRINT)

+ BALR 14,15

**A couple more macro instructions:**

**MEXIT**

This instruction terminates the processing of a macro definition.

Any instructions/statements that follow an MEXIT will NOT be part of the assembly listing.

MEXIT is usually used when error checking is being done within the macro body.

MACRO

&LABEL EXMPL1 &SUBRTN=BUILD,&PARMS=

&LABEL LA 1,&PARMS

L 15,=V(&SUBRTN)

MEXIT

BALR 14,15

MEND

The call of the EXMPL1 macro:

EXMPL1 SUBRTN=PRINT,PARMS=PARM2

In the assembly:

+ LA 1,PARM4

+ L 15,=V(PRINT)

**MNOTE**

This instruction is used to generate an error message in an assembly listing.

Format 1: MNOTE 'error message goes here'

If the MNOTE is executed, the error message will be displayed.

Format 2: MNOTE severity,'error message goes here'

This format of MNOTE assigns a severity code to the error. The severity code can be between 0 and 255 and defaults to 1 if it is not specified. The code is used to indicate if the message is an error or just a warning.

If this version of MNOTE is executed, the error message will be displayed along with the severity code.

**Conditional Assembly**

The sequence that the instructions/statements are processed in can be altered by using the conditional assembly instructions.

A **sequence symbol** is any symbol that satisifes the following rules:

* Two to eight characters in length
* The first character is a period
* The second character is a letter
* The remaining characters are either letters or digits

Sequence symbols are equivalent to a label in an assembler program.

**ANOP**

This instruction provides a label for other instructions to branch to. It is equivalent to a DS 0H in an assembler program.

Format: .sequence\_symbol ANOP

**AGO**

This instruction performs an unconditional branch.

Format: .sequence\_symbol\_1 AGO .sequence\_symbol\_2

An unconditional branch to .sequence\_symbol\_2 is taken. .sequence\_symbol\_1 is optional. If .sequence\_symbol\_1 is specified, it provides a label for another conditional assembly instruction to branch to.

**AIF**

This instruction performs a conditional branch.

Format: .seq\_sym\_1 AIF (conditional expression).seq\_sym\_2

A branch to .seq\_sym\_2 is taken if the result of the conditional expression is 1 (equivalent to TRUE).

If the conditional expression evaluates to 0 (equivalent to FALSE), the line immediately following the AIF is executed.

**AIF conditional expression**

Format: (operand\_1 relational\_operator operand\_2)

operand\_1 and operand\_2 may be:

* an arithmetic expression
* 0 or 1
* a character string enclosed in single quotes
* a variable symbol

relational\_operator may be:

* EQ for equality
* NE for not equal
* LT for less than
* LE for less than or equal to
* GT for greater than
* GE for greater than or equal to

For example:

AIF ('&PARMS' EQ '').NOPARM

If &PARMS is equal to the null string, branch to .NOPARM

MACRO

&LABEL EXMPL1 &SUBRTN=BUILD,&PARMS=

AIF ('&PARMS' EQ '').NOPARMS

&LABEL LA 1,&PARMS

.NOPARMS ANOP

L 15,=V(&SUBRTN)

BALR 14,15

MEND

The call of the EXMPL1 macro:

EXMPL1 SUBRTN=PRINT

In the assembly:

+ L 15,=V(PRINT)

+ BALR 14,15

MACRO

EQUREGS

LCLA &NUM

.LOOP AIF (&NUM GT 15).LPEND

R&NUM EQU &NUM

&NUM SETA &NUM+1

AGO .LOOP

.LPEND ANOP

MEND

The call of the EQUREGS macro:

EQUREGS

In the assembly:

+R0 EQU 0

+R1 EQU 1

...

...

+R15 EQU 15

MACRO

EXITLINK &TYPE

AIF ('&TYPE' NE '').FOUND

MNOTE '\*\*\* MISSING TYPE PARAMETER \*\*\*'

MEXIT

.FOUND ANOP

AIF ('&TYPE' EQ 'N').NORMAL

AIF ('&TYPE' EQ 'R').RETURN

AIF ('&TYPE' EQ 'V').VALUE

AIF ('&TYPE' EQ 'B').BOTH

MNOTE 8,'\*\*\* TYPE IS NOT VALID'

MEXIT

.NORMAL ANOP <--- NORMAL LINKAGE

L 13,4(,13)

LM 14,12,12(13)

BR 14

AGO .DONE

.RETURN ANOP <--- RETURN CODE IN REGISTER 15

L 13,4(,13)

L 14,12(,13)

LM 0,12,20(13)

BR 14

AGO .DONE

.VALUE ANOP <--- CALCULATED VALUE IN REGISTER 0

L 13,4(,13)

LM 14,15,12(13)

LM 1,12,24(13)

BR 14

AGO .DONE

.BOTH ANOP <--- VALUES IN REGISTER 15 AND 0

L 13,4(,13)

L 14,12(,13)

LM 1,12,24(13)

BR 14

.DONE ANOP

MEND

The call of the EXITLINK macro:

EXITLINK

In the assembly:

+ \*\*\* MISSING TYPE PARAMETER \*\*\*

Another call:

EXITLINK CHAR

+

+ L 13,4(,13)

+ L 14,12(,13)

+ LM 0,12,20(13)

+ BR 14

CHAR DC C'R'

**Data Attributes**

For each constant or instruction the assembler assigns **data attributes**. These attributes can be used with the conditional assembly instructions to control the sequence and contents of the statements generated.

**Length Attribute**

This attribute is a numeric value that is equal to the number of bytes occupied by the data that is represented by a symbolic parameter.

The value of the symbolic parameter MUST be a label in the calling program.

Format: L'symbolic\_parameter

MACRO

MOVEIT &DEST,&SOURCE

MVC &DEST.(L'&SOURCE),&SOURCE

MEND

In a program:

MOVEIT PLINE,TABLE

+ MVC PLINE(80),TABLE

MOVEIT PLINE,80

+ MVC PLINE(0),80

The length attribute is 0 because 80 is not a label in the calling

program.

MOVEIT PLINE,TABLE2

+ MVC PLINE(20),TABLE2

The length attribute is 20. 4 is the repetition factor so it is NOT

included in the length.

PLINE DS CL132

TABLE DS CL80

TABLE2 DS 4CL20

**Count Attribute**

This attribute is a numeric value that is equal to the number of characters in the actual parameter being passed.

If the value is 0, the parameter is missing.

Format: K'symbolic\_parameter

MACRO

PRINTIT &P1,&P2

LCLA &CNT1,&CNT2

AIF (K'&P1 NE 0).FOUND

MNOTE '\*\*\* MISSING PARAMETER \*\*\*'

MEXIT

.FOUND ANOP

&CNT1 SETA K'&P1

&CNT2 SETA K'&P2-2

XPRNT =C&P2,&CNT2

LA 5,&CNT1

MEND

In a program:

PRINTIT JUNKY,'1 TOP OF PAGE'

+ XPRNT =C'1 TOP OF PAGE',14

+ LA 5,5

K'&P1 --> K'JUNKY --> 5 since there are 5 letters in JUNKY

K'&P2-2 --> K''1 TOP OF PAGE'-2 --> 16-2 --> 14

the 2 is subtracted to account for the single quotes

PRINTIT WHATEVER,'0 DOUBLE SPACE'

+ XPRNT =C'0 DOUBLE SPACE',15

+ LA 5,8

**Number Attribute**

This attribute is a numeric value that is equal to the number of operands in an operand sublist.

If this attribute is used on &SYSLIST, will give the number of positional parameters on the prototype statement.

If this attribute is used on &SYSLIST(m), will give the number of items in the sublist of the mth parameter.

If this attribute is used on any other operand, will give the number of items in the sublist of that parameter.

Format: N'symbolic\_parameter

The macro that follows is going to generate a branch table similar

to one used in the hashing assignment. &BTAB is a sublist of the

different labels to branch to. The first member is for a return code

of 0, the second a return code of 4, etc. If &BTAB is not specified

when the macro is called, do not generate the table. If &BTAB is

specified, you may assume that there is at least one member in the

sublist. The table that is generated should have a unique label

(BTAB????) in case the macro is called more than one time. After all

of the necessary branches have been created, a SOC1???? DC H'0'

should be coded so that the program abends if an improper return code

is returned in register 15.

MACRO

BRANCHTAB &BTAB

LCLC &NDX

LCLA &NUM,&CNT

&NDX SETC '&SYSNDX'

&CNT SETA 2

&NUM SETA N'&BTAB

AIF ('&BTAB' EQ '').NOTAB

B BTAB&NDX.(15)

BTAB&NDX B &BTAB(1)

.BLOOP AIF (&CNT GT &NUM).DONE

B &BTAB(&CNT)

&CNT SETA &CNT+1

AGO .BLOOP

.DONE ANOP

SOC1&NDX DC H'0'

.NOTAB ANOP

MEND

In a program:

BRANCHTAB (RC0,RC4,RC8)

+ B BTAB0001(15)

+BTAB0001 B RC0

+ B RC4

+ B RC8

+SOC10001 DC H'0'

**Type Attribute**

This attribute indicates the type of data of the field assigned to the symbolic parameter when the macro is called.

Format: T'symbolic\_parameter

Symbols for DC/DS statement Meaning

A Address constant

B Binary constant

C Character constant

F Fullword

H Halfword

P Packed decimal

R A-con or V-con

X Hexadecimal

Z Zoned decimal

V V-con

Symbols for other statement Meaning

I Machine Instruction

J CSECT name

M Macro Instruction

O Omitted Operand

MACRO

TEXMPL &P1,&P2

LCLC &CH1,&CH2

AIF (T'&P1 NE 'O').FOUND

MNOTE '\*\*\*\* PARAMETER IS MISSING \*\*\*\*'

MEXIT

.FOUND ANOP

&CH1 SETC T'&P2

&CH2 SETC T'&P1

&CH2 SETC '&CH2&CH1'

DC C'&CH2'

MEND

In a program:

TEXMPL TABLE,PLINE

+ DC C'FC'

TABLE DS 20F

PLINE DC CL12