Macro

Introduction

- A macro instruction (abbreviated to macro) is simply a notational convenience for the programmer.
- Represents a commonly used group of statements in the source programming language
- Expanding a macros
  - Replace each macro instruction with the corresponding group of source language statements
- We will follow SIC (Simplified Instructional Computer) architecture

Introduction

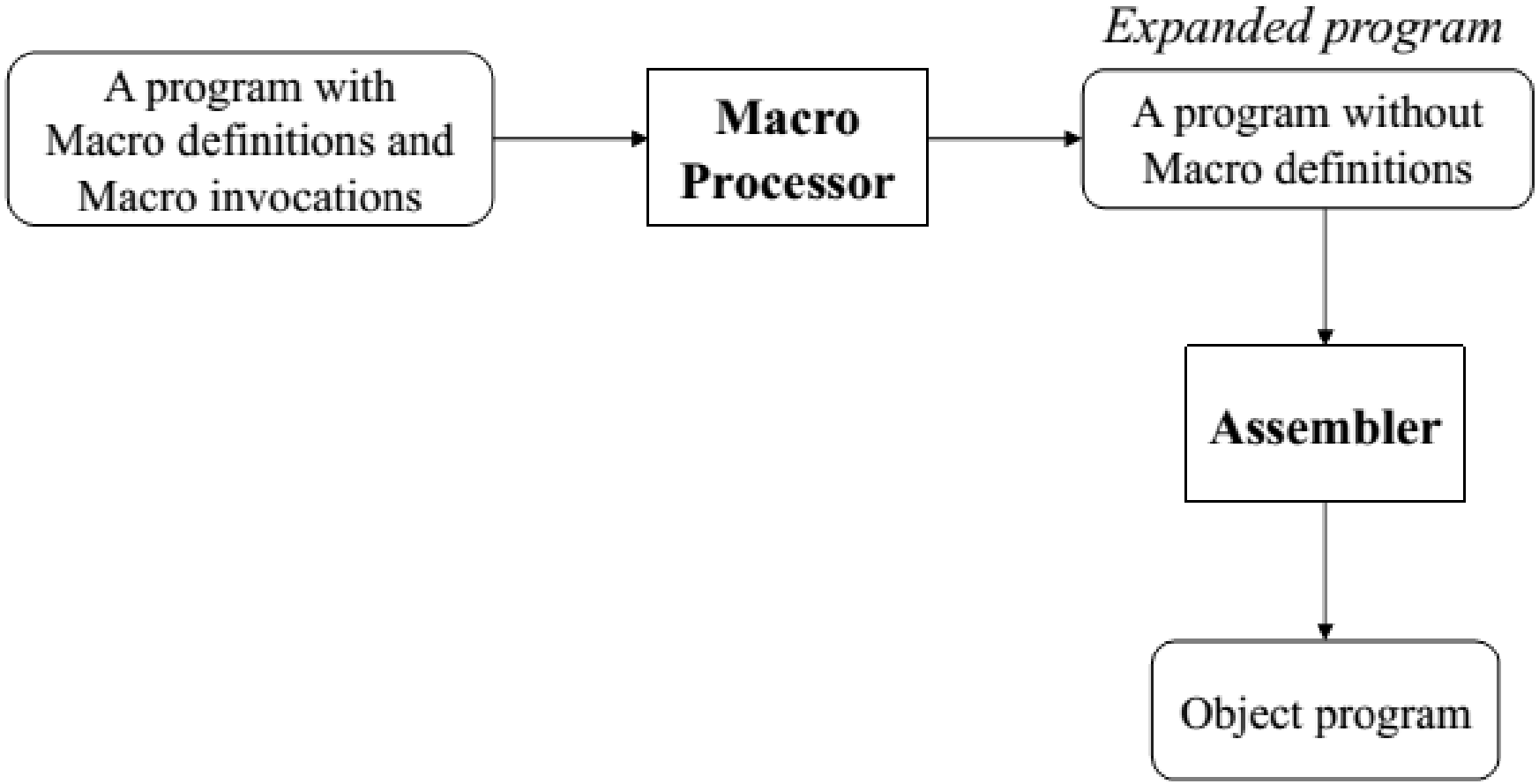
- SIC:
  - 24 bit registers
  - A, X, L - (0,1,2)
  - PC
  - SW (Status Word) - (8,9)

  - GPR:
    - B, S, T, F - (3,4,5,6)

Introduction

- For example:
- On SIC/XE (Extra Equipment), requires a sequence of seven instructions to save the contents of all registers
  - Write one statement
  - like SAVERGS
- A macro processor is not directly related to the architecture of the computer on which it is to run
- Macro processors can also be used with high-level programming languages, OS command languages, etc.

Basic Macro Processor Functions



Basic Macro Processor Functions

Macro Definition:

1. Two new assembler directives
   - **MACRO**
   - **MEND**
2. A pattern or **prototype** for the macro instruction
3. Macro name and parameters

Basic Macro Processor Functions

- Macro invocation
  - Often referred to as a ***macro call***
  - Need:

  a) name of the macro instruction being invoked

  b) arguments to be used in expanding the macro
- Expanded program
  - No macro instruction definitions
  - Each macro invocation statement has been expanded with

  a) Macro body

  b) Arguments substituted with the parameters in the prototype

Macro Definition

1. Copy code
2. Parameter substitution
3. Conditional macro expansion
4. Macro instruction defining macros

1. Copy Code

Source

```
STRG    MACRO
        STA     DATA1
        STB     DATA2
        STX     DATA3
        MEND
   .
STRG
   .
STRG
   .
   .
```

Expanded source

```
   .
   .
   .
      STA     DATA1
      STB     DATA2
      STX     DATA3
   .
      STA     DATA1
      STB     DATA2
      STX     DATA3
   .
```

2. Parameter Substitution

Source

```
STRG    MACRO   &a1, &a2, &a3
        STA     &a1
        STB     &a2
        STX     &a3
        MEND
  .
STRG    DATA1, DATA2, DATA3
  .
STRG    DATA4, DATA5, DATA6
  .
  .
```

Expanded source

```
  .
  .
  .
    STA     DATA1
    STB     DATA2
    STX     DATA3
  .
    STA     DATA4
    STB     DATA5
    STX     DATA6
  .
```

Parameter Substitution

- Dummy arguments
  - Positional argument

```
STRG    DATA1, DATA2, DATA3
GENER   ,,DIRECT,,,,,,3
```

  - Keyword argument

```
STRG    &a3=DATA1, &a2=DATA2, &a1=DATA3
GENER   TYPE=DIRECT, CHANNEL=3
```

Example

Line **Source statement**

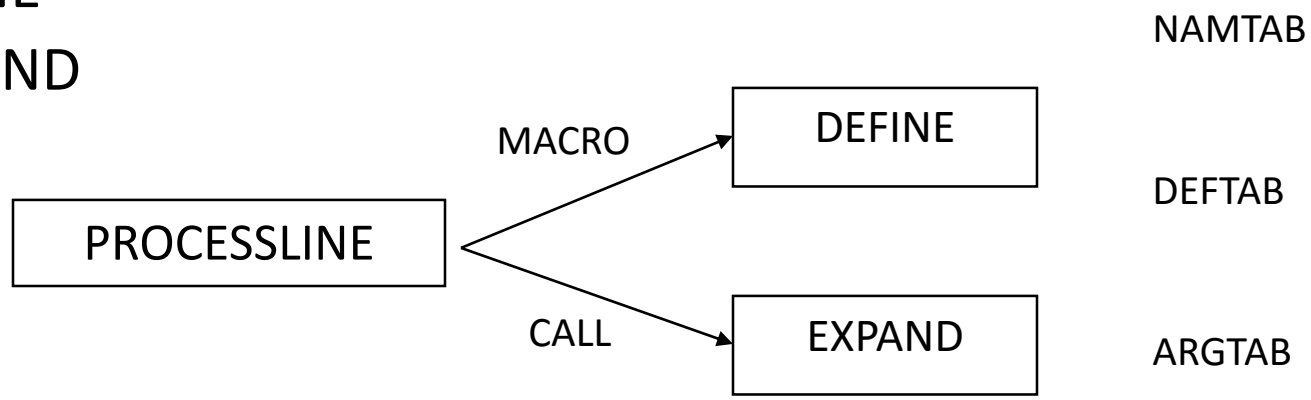
```
 5     COPY     START   0                 COPY FILE FROM INPUT TO OUTPUT
10     RDBUFF   MACRO   &INDEV,&BUFADR,&RECLTH
15     .
20     .        MACRO TO READ RECORD INTO BUFFER
25     .
30              CLEAR   X                 CLEAR LOOP COUNTER
35              CLEAR   A
40              CLEAR   S
45              +LDT    #4096             SET MAXIMUM RECORD LENGTH
50              TD      =X'&INDEV'        TEST INPUT DEVICE
55              JEQ     *-3               LOOP UNTIL READY
60              RD      =X'&INDEV'        READ CHARACTER INTO REG A
65              COMPR   A,S               TEST FOR END OF RECORD
70              JEQ     *+11              EXIT LOOP IF EOR
75              STCH    &BUFADR,X         STORE CHARACTER IN BUFFER
80              TIXR    T                 LOOP UNLESS MAXIMUM LENGTH
85              JLT     *-19                HAS BEEN REACHED
90              STX     &RECLTH           SAVE RECORD LENGTH
95              MEND
```

Example

```
100     WRBUFF    MACRO    &OUTDEV,&BUFADR,&RECLTH
105     .
110     .         MACRO TO WRITE RECORD FROM BUFFER
115     .
120               CLEAR    X                CLEAR LOOP COUNTER
125               LDT      &RECLTH
130               LDCH     &BUFADR,X        GET CHARACTER FROM BUFFER
135               TD       =X'&OUTDEV'      TEST OUTPUT DEVICE
140               JEQ      *-3              LOOP UNTIL READY
145               WD       =X'&OUTDEV'      WRITE CHARACTER
150               TIXR     T                LOOP UNTIL ALL CHARACTERS
155               JLT      *-14               HAVE BEEN WRITTEN
160               MEND
```

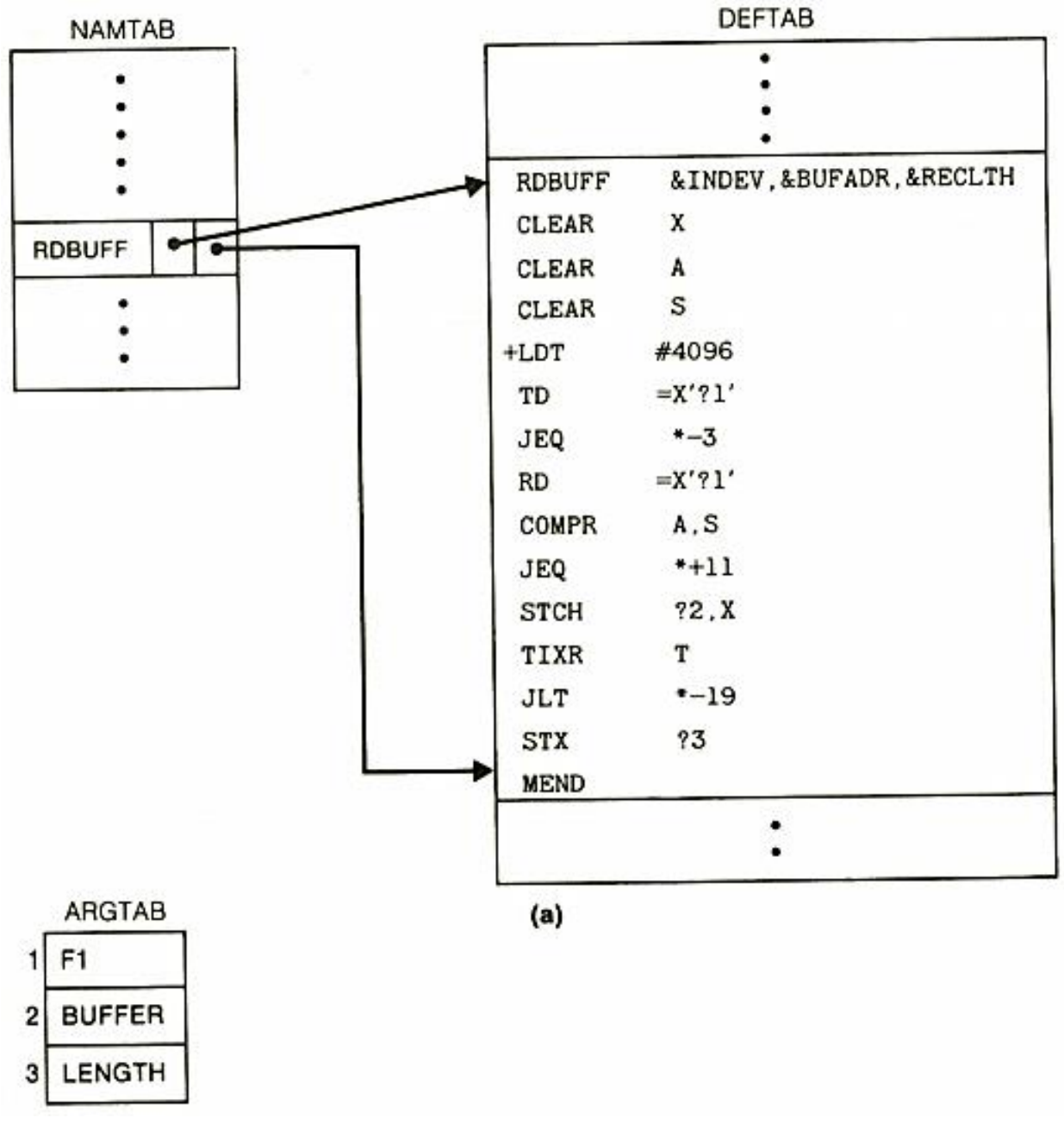
One Pass Macro Processor

- Prerequisite
  - every macro must be defined before it is called
- Sub-procedures
  - macro definition: DEFINE
  - macro invocation: EXPAND



Data Structures

- Because of the one-pass structure, the definition of a macro must appear in the source program before any statements that invoke that macro
- Three main data structures involved in an one-pass macro processor
  - DEFTAB,
  - NAMTAB,
  - ARGTAB


NAMTAB
RDBUFF
DEFTAB
RDBUFF &INDEV,&BUFADR,&RECLTH
CLEAR X
CLEAR A
CLEAR S
+LDT #4096
TD =X'?1'
JEQ *-3
RD =X'?1'
COMPR A,S
JEQ *+11
STCH ?2,X
TIXR T
JLT *-19
STX ?3
MEND
(a)
ARGTAB
1 F1
2 BUFFER
3 LENGTH

```
begin {macro processor}
    EXPANDING := FALSE
    while OPCODE ≠ 'END' do
        begin
            GETLINE
            PROCESSLINE
        end {while}
end {macro processor}


procedure PROCESSLINE
    begin
        search NAMTAB for OPCODE
        if found then
            EXPAND
        else if OPCODE = 'MACRO' then
            DEFINE
        else write source line to expanded file
    end {PROCESSLINE}
```

Figure 4.5 Algorithm for a one-pass macro processor.

Nested Macro

```
1   MACROS    MACRO       {Defines SIC standard version macros}
2   RDBUFF    MACRO       &INDEV,&BUFADR,&RECLTH
                .
                .         {SIC  standard version}
                .
3               MEND      {End of RDBUFF}
4   WRBUFF    MACRO       &OUTDEV,&BUFADR,&RECLTH
                .
                .         {SIC standard version}
                .
5               MEND      {End of WRBUFF}
                .
                .
                .
6               MEND      {End of MACROS}
```

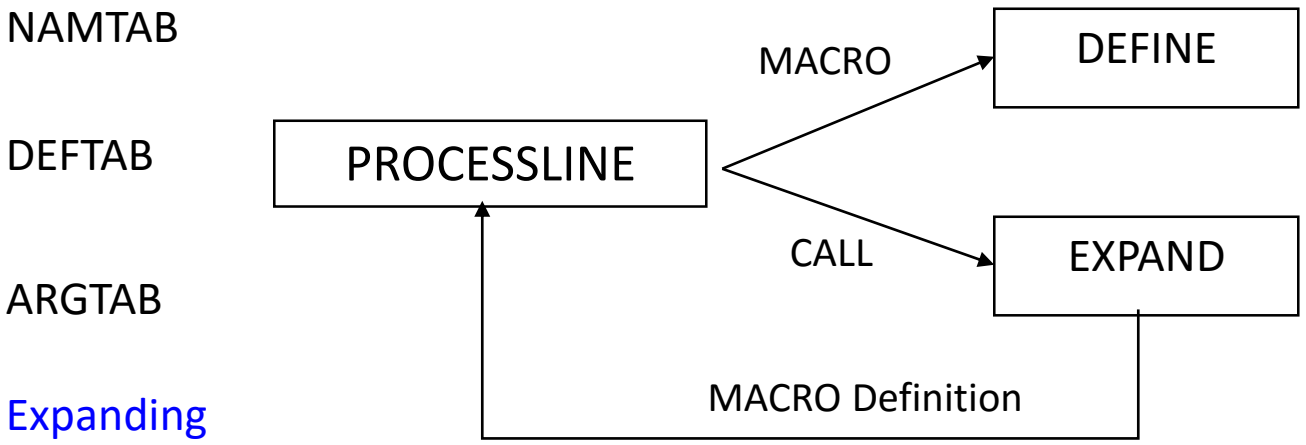
(a)

Nested Macro

```
1   MACROX    MACRO      {Defines  SIC/XE macros}
2   RDBUFF    MACRO      &INDEV,&BUFADR,&RECLTH
                .
                .        {SIC/XE version}
                .
3               MEND     {End of RDBUFF}
4   WRBUFF    MACRO      &OUTDEV,&BUFADR,&RECLTH
                .
                .        {SIC/XE version}
                .
5               MEND     {End of WRBUFF}
                .
                .
                .
6               MEND     {End of MACROX}
```

One Pass Macro Processor:
for nested macro

- Sub-procedures
  - macro definition: DEFINE
  - macro invocation: EXPAND
- EXPAND may invoke DEFINE when encounter macro definition



```
procedure DEFINE
    begin
        enter macro name into NAMTAB
        enter macro prototype into DEFTAB
        LEVEL  := 1
        while LEVEL > 0 do
            begin
                GETLINE
                if this is not a comment line then
                    begin
                        substitute positional notation for parameters
                        enter line into DEFTAB
                        if OPCODE = 'MACRO' then
                            LEVEL := LEVEL + 1
                        else if OPCODE = 'MEND' then
                            LEVEL  := LEVEL - 1
                    end {if not comment}
            end {while}
        store in NAMTAB pointers to beginning and end of definition
    end {DEFINE}
```

```
procedure EXPAND
    begin
        EXPANDING := TRUE
        get first line of macro definition {prototype} from DEFTAB
        set up arguments from macro invocation in ARGTAB
        write macro invocation to expanded file as a comment
        while not end of macro definition do
            begin
                GETLINE
                PROCESSLINE
            end {while}
        EXPANDING := FALSE
    end {EXPAND}
```

```
procedure GETLINE
    begin
        if EXPANDING then
            begin
                get next line of macro definition from DEFTAB
                substitute arguments from ARGTAB for positional notation
            end {if}
        else
            read next line from input file
    end {GETLINE}
```

Figure 4.5 (*cont'd*)

One Pass Macro Processor