SLAC Computation Group
Stanford, California

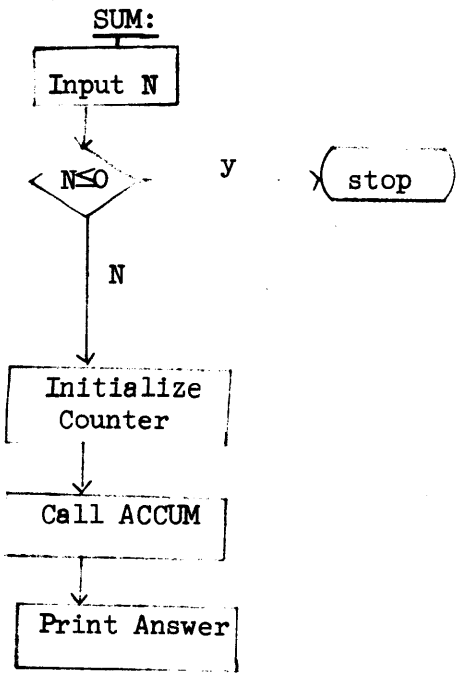
CGTM 140
Bary W. Pollack
October 1972

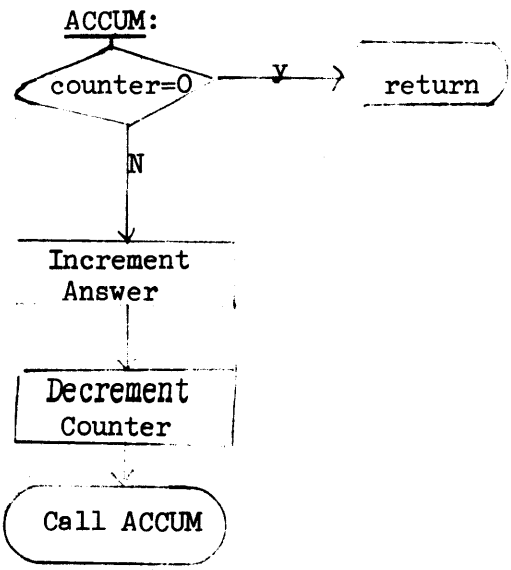
RECURSIVE WYLBUR EXEC FILES

The WYLBUR command structure includes the facilities necessary for EXEC files to call one another. This note illustrates a technique for writing EXEC files that not only call each other, but return as well - an ability which (for reasons best known only to the system implementers) has been left out of the WYLBUR language. This technique is completely general. As an illustration, we present a recursive WYLBUR program to compute $\sigma(n)$. The sum of all integers less than or equal to n:

n	$\sigma(n)$
0	0
1	1
2	3
3	6
4	10
5	15
6	21

We give flow charts for the two routines, SUM and ACCUM:





These two programs compute $\sigma(n)$ according to the recursive definition:

$$\sigma(n) = \begin{cases} \text{if } n=0, \ 0 \\ \text{if } n>0, \ n+\sigma(n-1) \end{cases}$$

In the discussion which follows, it will be helpful to refer to the WYLBUR programs given at the end of this memo.

We have made the arbitrary restriction that a recursive program may not use the Active file. To get around this restriction, one need only save and restore the Active file before and after each CALL or RETURN. We use the Active file as a pushdown stack. The last line always contains a reference to the most recent return point.

CALL pushes the return point onto the stack and executes the called program.

RETURN pops the stack and executes the program specified by the return point.

INIT sets up the stack and asks for the name of the (recursive) function to be executed.

FINI cleans things up after execution is finished.

ASM is an assembler which accepts WYLBUR files and expands all CALLs and RETURNs into the appropriate code. Input files must be capable of being renumbered. One normally would use ASM first, and then the assembler, ASMEXEC, which resides in the user library on WYL001. ASM accepts a series of files and assembles them. Each is output to the disk with its name suffixed with an R. E.g., if FOO is the name of a file input to ASM, FOOR will be the name of the output file. This series of files may be terminated by typing * which means "clean things up and quit", or ** which means "cleans things up and call INIT".

To run this example, the files ASM, INIT, FINI, CALL, RETURN, SUM, ACCUM should all be in your library.

```
EXEC FROM #ASM  (initates the assembler)
(Assemble #SUM and #ACCUM)
(Terminate with **)
(Respond #SUM when INIT asks for the function)
(SUM is now running, give it an integer, n, it will
 complete σ(n). Repeat as many times as you wish.
 Finally, give SUM an integer ≤ 0)
(Execution ends)
```

The program listings follow.

```
1.          ;  #SUM
2.          ;
3.          ;  THIS ROUTINE RECURSIVELY ACCUMULATES THE SUM
4.          ;  OF ALL NUMBERS LESS THAN OR EQUAL TO THE VALUE
5.          ;  IN NO.  IT USES N1 TO STORE THE PARTIAL RESULT.
6.          ;
7.          COMMENT
8.          READ PRO 'SUM ALL NUMBERS LESS THAN OR EQUAL TO:  ' VAL NO
9.          IF (NO LE 0) EXE 14
10.         SET VAL N1 0
11.         CALL #ACCUM
12.         COMMENT THE VALUE IS &N1
13.         EXE 7
14.         RETURN
```

```
1.          ;  #ACCUM
2.          ;
3.          ;  THIS ROUTINE RECURSIVELY CALLS ITSELF TO
4.          ;  ACCUMULATE THE SUM.  IT STORES ITS PARTIAL
5.          ;  RESULT IN N1.
6.          ;
7.          IF (NO EQ 0) EXE 11
8.          SET VAL N1 &NO+&N1
9.          SET VAL NO &NO-1
10.         CALL #ACCUM
11.         RETURN
```

```
1.          ;  #INIT
2.          CLE ACT
3.          SET ESC &
4.          1 1          #FINI
5.          COMMENT
6.          READ PRO 'BEGIN FUNCTION:  ' STR S9
7.          2 1          &S9R
8.          COMMENT
9.          EXE PRO &S9R CLE
```

```
1.          ;  #FINI
2.          SET ESC
3.          SET VAL S8 ''
4.          SET VAL S9 ''
5.          SET VAL N9 0
6.          COMMENT
7.          COMMENT
8.          COMMENT END EXECUTION
9.          COMMENT
10.         CLE EXE NAM ACT
```

```
1.      SET VAL S9 SUBSTR(*CUR*||'          ',1,9)
2.      CH 1/9 TO '&S9' IN L N
3.      END *FN*
4.      EXE FRO *FN* CLE
```

```
1.      SET VAL S8 S9
2.      DEL L
3.      READ VAL N9 USING L COLS 1/5
4.      READ STR S9 USING L COLS 10/15
5.      IF (S8 EQ S9) EXE &N9
6.      EXE FRO &S9 START &N5 CLE
```

```
1.                ;  #ASMS  --  RECURSIVE PROCEDURE ASSEMBLER
2.                ;
3.                ;  ASSEMBLES UNTIL GIVEN A '*'  --  QUIT
4.                ;                          OR '**'  --  CALL INIT
5.                ;  FILES MUST BE RE-NUMBERABLE
6.                ;
7.                SET ESC &
8.       LBL00    COMMENT
9.                READ PRO 'FILE?  ' STR SO
10.               IF (SO EQ '*') EXE LBL05
11.               IF (SO EQ '**') EXE LBL04
12.               USE &SO CLE
13.               NUM
14.      LBL01    P 'CALL ' (1) N
15.               IF (* EQ -1) EXE LBL02
16.               SET VAL NO *
17.               CH 'CALL ' TO '' IN * N
18.               READ STR S9 USING *
19.               DEL *
20.               COPY ALL TO &NO FROM #CALL BY .001
21.               IF (SO EQ SUBSTR(S9,INDEX(S9,'#'))) * EXE 1
22.               CH '*FN*' TO '&S9R' N
23.               CH '*CUR*' TO '&NO+1' N
24.               EXE LBL01
25.      LBL02    P 'RETURN ' (1) N
26.               IF (* EQ -1) EXE LBL03
27.               SET VAL NO CUR
28.               DEL *
29.               COPY ALL TO &NO FROM #RETURN BY .001
30.               EXE LBL02
31.      LBL03    SAV &SOR REP CLE
32.               EXE LBL00
33.      LBL04    SET VAL SO ''
34.               SET VAL S9 ''
35.               SET VAL NO 0
36.               EXE FRO #INIT CLE
37.      LBL05    EXE FRO #CLEANUP CLE
```