(iv) Writing Assembler Language Subprograms

This section describes the conventions to be used when coding subprograms in assembler language for use with WATFIV. The conventions are, in fact, similar to those required by the G/H compilers, i.e., an experienced assembly language programmer should have little trouble converting existing routines to run under WATFIV.

Symbolic notation is used for registers throughout the following description; the naming convention is the obvious one.

1. Subprogram Calling Sequences

Suppose a subroutine or function subprogram 'rtn' is referred to by a CALL statement or function reference in a FORTRAN source statement, e.g.,

```
(i)   CALL rtn (arg1,arg2,arg3,...,argn)
(ii)  Y=rtn(arg1,arg2,arg3,...,argn)
```

The calling sequence generated by WATFIV for either case is

```
                CNOP    2,4
                LA      R14,return
                L       R15,=V(rtn)
                BALR    R1,R15
                DC      AL1(c1),AL3(addr1)
                DC      AL1(c2),AL3(addr2)
argument        .
list            .
                .
                DC      AL1(cm),AL3(addrm)
return          EQU     *
```

VIL4 SEP./69

Each ci is a code which describes the kind of argument list entry; each addri is either the address of an argument or a pointer to more information about the argument. The calling routine also provides an 18 word OS-type save area.

Thus, on entry to the called subprogram

- R15 contains the address of the entry point

- R14 contains the normal return address

- R13 contains the address of a save area

- R1 contains the address of an argument list aligned on a word boundary.

Moreover, it is a WATFIV convention that F6 will contain zero and R12 will contain the base address of a set of routines, constants, switches, etc. internal to the compiler.

When control is returned to the WATFIV-compiled program, it expects that:

- at least its registers R5-R13 have been restored

- the result of a function reference is returned in

  (a) R0 if the function is integer or logical

  (b) F0 if the function is real

  (c) (F0,F2) if the function is complex.

- F6 still contains zero.

The six major categories for a code byte ci and associated meanings of addri are:

(a) Unchangeable quantity - Q: ci=B'0000mmmm'

An argument which should not be changed by the called subprogram is flagged with this code byte. Unchangeable quantities are constants, temporaries, DO-parameters, assigned GOTO indices.

The low order four bits mmmm give the type of the argument according to Table 1.

For type numbers 0-7, AL3(addri) = AL3(Q); for type numbers 8 and 9, AL3(addri)=AL3(Q*) where Q* is a full-word of the form DC AL1(n),AL3(Q).

| Type | TYPE NUMBER | mmmm | s-VALUE |
|---|---|---|---|
| Logical*4 | 0 | 0000 | 2 |
| Logical*1 | 1 | 0001 | 0 |
| Integer*4 | 2 | 0010 | 2 |
| Integer*2 | 3 | 0011 | 1 |
| Real*4 | 4 | 0100 | 2 |
| Real*8 | 5 | 0101 | 3 |
| Complex*8 | 6 | 0110 | 3 |
| Complex*16 | 7 | 0111 | 4 |
| Character*n n=1 | 8 | 1000 | 0 |
| Character*n n>1 | 9 | 1001 | 0 |

TABLE 1. Type-code Bits

(b) Variables, Array Elements - V: ci='B1000mmmm'

Here, AL3(addri) = AL3(V), and mmmm is given in Table 1. If the argument is an array element, an extra word follows in the argument list as a special indicator. This has the form

DC X'8C',AL3(V*)

where V* is the, so-called, star routine for the array of which the element is a member. Star routines will be described below.

Except that for CHAR*N, AL3(addri) = AL3(V*), when V* is AL1(n),AL3(V) (as for (a) above).

For example, if A(5) is used as a subprogram argument, the corresponding argument list entry would appear as follows:

```
DC   B'10000100',AL3(A1+16),X'8C',AL3(A*)
```

where A1 is used as symbol to represent A(1). (A is assumed to be REAL*4).

(c) Array name- A ci=B'1kkkmmmm'

Here, kkk is the number of dimensions of A, and

AL3(addri) = AL3(A*)

(d) Subprogram name- R

If R is a subroutine, ci=B'01010000'; if R is a function, ci=B'0110mmmm'. In both cases AL3(addri) = AL3(R*) where R* is DC A(R).

(e) Statement number - &n ci=B'00110000'

Here AL3(addri) = AL3(n*) where n* is the address of the statement numbered n.

(f) Argument list terminator

This is a special entry to mark the end of the argument list; it also provides information about the nature of the called routine.

If the called routine is to be a subroutine, ci=B'00010000'; if the called routine is to be a function, ci=B'0010mmmm'. The accompanying adcon contains no information.

## 2. Star Routines for Array Arguments

All references to an array, say X, in a WATFIV-compiled program are made by means of the Subscript Testing and Addressing Routine (or star routine), for X, called X*. The star routines for all arrays declared in FORTRAN source programs are constructed by the compiler. Each star routine contains information pertinent to an array (e.g., its dimensions, starting address, total length), and is used at execution time for indexing into the array, for checking for out-of-range subscripts, and for passing the array to subprograms.

Knowledge of the form of a star routine is required when an assembler subprogram must receive (or pass) an array from (or to) a higher-level (or lower-level) FORTRAN routine. For these purposes, it is sufficient to consider a 'skeleton' star routine, say X*, of the following form.

```
        DS    0F
X*      EQU   *-4
        DC    AL1(f),AL3(1st element in the array)
        DC    AL1(s),AL3(length, in bytes, of the array)
```

Here, f = 4k-4 where k is the number of dimensions
s = s-value corresponding to the type of the array (see TABLE 1).

In a star routine constructed to pass an array to a FORTRAN subprogram, the f and s bytes may be set to zero; the prologue in the FORTRAN routine references only the two AL3 adcons when initializing the star routine of the dummy array to which the calling array is passed.

The form of a full star routine constructed by the compiler is given below in section (4) for documentation purpose.

## 3. Other Conventions for Assembler Subprograms

(a) Only the first 6 characters of CSECT, ENTRY, EXTRN names are used by WATFIV's object deck loader; names longer than 6 characters are simply truncated. Names of ENTRY points and CSECTS must be unique; i.e., a deck may not have a csect whose first 6 characters are the same as an entry point in that deck.

(b) Blank COMMON may be referred to by the usual COM assembler feature. To refer to named COMMON, the V-type address constant name DC V(name of COMMON) is used.

(c) Assembler subprograms may use the CXD and DXD assembler features.

(d) A logical function returns its value in the low-order byte of R0; .TRUE. is X'FF', .FALSE. is X'00'. WATFIV stores the value of a logical variable in the high-order byte.

(e) To simulate a multiple return statement (RETURNi), the called subprogram must search the argument list for the ith statement number argument. The address to which control should be returned can be determined from this argument list entry. See section (l) above.

(f) To call a FORTRAN subprogram from an assembler subprogram, do the following:

- simulate a WATFIV-generated call as described in section (i). This will include a properly constructed argument list and any required skeleton star routines. Provide a save area address in R13.

- pass on the contents of R12 that were passed to the assembler subprogram by a higher-level FORTRAN routine.

- ensure that F6 is zeroed.

- upon return from the FORTRAN subprogram, restore the assembler subprogram's registers R0-R4, R12, R15, if required, from its save area. The FORTRAN subprogram's epiloque restores only R5-R11, R13-R14. (Function values are returned in R0, F0 or F0-F2 as described in section (l).) The special precautions for registers R12 and F6 are required since WATFIV assumes that they remain constant throughout the execution of a program. F6 is used for converting single precision values to double precision; R12 is used as a base register for many internal execution time routines contained in compiler csect STARTA. Given below is an example to show the above linkage. Note that the save areas are chained using the standard OS rules.

(g) To issue error message with a trace-back and the subroutine's name in the error-message (if the name usually appears in message) issue:

```
ERROR (NOAC,XX,#,ENTreg),XRETRACE
```

where XX,# is the error message number

reg is a register that points to the subroutine's name.

Example: to issue SR-4 in subroutine SUBPRO issue:

```
ERROR (NOAC,SR,4,ENTR11),XRETRACE
```

where earlier in the subprogram R11 has been loaded as follows:

```
LA R11,NAME
```

and NAME is defined as:

```
NAME  DC  CL6'SUBPRO'
```

This is a sample WATFIV program to demonstrate some conventions used when coding subprograms in assembler language for use with WATFIV.

The main program calls an assembler language subprogram RTN. RTN in turn calls the WATFIV written subprogram NEXT. RTN passes a value for XX of 12.25 to be used in NEXT. NEXT initializes the array B and the variable Y and returns. RTN then passes back to the main program a value for X.

```
      COMMON I
      DIMENSION A(10)
      CALL RTN(A,X)
      PRINT,A,X,I
      STOP
      END
      SUBROUTINE NEXT(B,Y)
      COMMON J
      DIMENSION B(5)
      PRINT,'HELLO FROM NEXT',' Y=',Y
      DO 1 J=1,5
    1 B(J)=J
      Y=-17.5
      PRINT,'GOOD-BYE FROM NEXT',' Y=',Y
      RETURN
      END
```

```
         START
         ENTRY     RTN
         USING     RTN,15
RTN      STM       14,12,12(13)        SAVE CALLER'S REGISTERS.
         LR        2,13                ADDR OF CALLER'S SAVE AREA.
         LA        13,SAVE             NEW SAVE AREA.
         ST        13,8(2)             LINK THE TWO
         ST        2,SAVE+4            SAVE AREAS.
         L         2,0(1)              ADDRESS OF 'STAR' ROUTINE FOR 'A'
         L         3,4(2)              ADDRESS OF 1ST ELEMENT OF 'A'
         LA        3,0(3)              GET RID OF CODE BYTE
         ST        3,ASTAR+4           SAVE IN DUMMY 'STAR' ROUTINE.
         L         3,8(2)              LENGTH of ARRAY 'A'.
         ST        3,ASTAR+8           SAVE in DUMMY 'STAR' ROUTINE.
         L         2,4(1)              ADDRESS OF SECOND ARGUMENT 'X'
         ST        2,XADDR             SAVE IT FOR LATER.
         LA        1,ARGLIST           ADDRESS OF ARGUMENT LIST FOR CALL
         L         15,=V(NEXT)         TO FORTRAN ROUTINE 'NEXT'
         BALR      14,15               AND AWAY WE GO.....
         DROP      15
         USING     *,14
         L         2,XADDR             ADDR 'X' PASSED FROM MAIN PROG.
         MVC       0(4,2),XX           RETURN VALUE FOR 'X' IN MAIN PROG
         L         13,SAVE+4           OLD SAVE AREA POINTER.
         LM        14,12,12(13)        MAIN PROGRAM'S REGISTERS.
         BR        14                  RETURN TO MAIN PROGRAM.
XX       DC        E'12.25'            CHANGED BY'Y=-17.5' IN 'NEXT'
*        ARGUMENT LIST FOR THE CALL TO 'NEXT' I.E. CALL NEXT(A,XX)
ARGLIST  DC        B'10010100',AL3(ASTAR)  POINTER TO STAR ROUTINE'A'
         DC        B'10000100',AL3(XX)     POINTER TO 'XX'.
         DC        B'00010000',AL3(0)      END OF LIST INDICATOR.
XADDR    DC        A(*-*)              SAVE ADDR OF 'X' HERE.
SAVE     DS        18F                 SAVE AREA.
ASTAR    EQU       *-4                 THIS IS A STAR ROUTINE TO
         DC        2A(*-*)             PASS 'A' TO ROUTINE 'NEXT'
         END
```

## 4. Compiler-Generated Star Routines

The form of a full star routine generated by the compiler for an array A is as follows:

```
        CNOP    2,4
        DC      CL6'A'
A*      BAL     R15,xrtn          See note 1
        DC      AL1(f),AL3(1st element in array)
        DC      AL1(s),AL3(length, in bytes, of array)   See note 2
        DC      A(n)              See note 3
        DC      A(d1)
        DC      A(d2)
                                  only k present
        .
        .
        .
        DC      A(dk)
        DC      B'C0C1C2C3C4C5C6C7',AL3(a1)
        DC      B'C00000000',AL3(a2)
        DC      B'C00000000',AL3(a3)                     See note 4
        .
        .
        .
        DC      B'C00000000',AL3(aj)
```

Here, k is the number of dimensions declared for A,

j is the number of variable dimensions, (maximum of 7)

d1, d2, etc., are the dimensions to be used for calculating the position of an element within the array,

f = 4k-4,

s = s-value from TABLE 1.

NOTES:

1. The symbol 'xrtn' stands for XA1 if k=1 or XAN if k>1. XA1 and XAN are names of subscript evaluation routines internal to the compiler, and are contained in compiler csect STARTA, addressable by R12. If the array is of type CHARACTER*n with n>1, the instruction is formed with R12 in the index position instead of the base position, i.e., xrtn is d(R12) instead of d(,R12). This fact is used by the error editor ERROR.

2. If the array is a dummy in a subprogram, when the subprogram is called, the prologue fills in the adcon for the first element from information supplied by the corresponding actual argument. Similarly, the prologue computes and fills in the length adcon if the dummy array has any variable dimensions.

3. This word is present only if the array is of type CHARACTER*n with n>1; in this case, f=4k. In fact, the compiler treats such arrays as if they had k+1 dimensions where the implied first dimension is n.

4. The first word is present if the array is a dummy array in a subprogram. Bits C1 to C7 indicate which, if any dimensions are variable; C1 is 1 if the last dimension is variable; C2 is 1 if the second last dimension is variable, and so on. If none of C1 to C7 are 1, then both C0 and a1 are zero; otherwise, C0 and a1 specify the location of the last dimension which is variable, as follows:

- if C0 = 0, then AL3(a1) = AL3 (variable dimension)

- if C0 = 1, then AL3(a1) = AL3($\overline{v}$) where $\overline{v}$ is defined by DC A (variable dimension)

  Bit C0 will be 1 if the variable dimension is in COMMON or is a call-by-location subprogram parameter. The second, third, etc., words are present if there are two, three, etc., variable dimensions; the second word locates the second last dimension which is variable, the third word locates the third last dimension which is variable, and so on. For these words, C0 and a2, a3, ..., are interpreted as above.

The set of words described in note 3 are used at execution time by the subprogram prologue to fill in the corresponding values in the list of dimensions, and to compute the total length of the array. The prologue fills

in star routines for dummy arrays only after it has passed down all other variables.

Example:

For source statements:

```
      SUBROUTINE RTN(ALPHA,X,/M/,N)
      COMMON K
      DIMENSION ALPHA(10,K,N,5,M,12)
```

the compiler constructs the following star routine for ALPHA.

```
        CNOP    2,4
        DC      CL6'ALPHA'
ALPHA*  BAL     R15,XAN
        DC      AL1(20),AL3(*-*)    1st element; filled by prologue
        DC      AL1(2),AL3(*-*)     length; filled by prologue
        DC      A(10)
        DC      A(*-*)              filled by prologue from K
        DC      A(*-*)              filled by prologue from N
        DC      A(5)
        DC      A(*-*)              filled by prologue from M
        DC      A(12)
        DC      B'10101100',AL3(M̄)
        DC      B'00000000',AL3(N̄)
        DC      B'10000000',AL3(K̄)
```

The compiler also constructs $\overline{M}$ and $\overline{K}$ as

```
K̄       DC      A(K)
M̄       DC      A(*-*)  filled in by prologue.
```

A star routine is stored in the local data area of the program segment in which it is declared; storage for an array which is not a dummy is allocated in the 'array area'.