516-50
EPR/RFG
7/12/71

USE OF THE 516 SEGMENT ASSEMBLER'S MACROS IN APPLICATION PROGRAMS

Definitions, calling sequences, remarks and examples of some 516 Segment Assembler's MACROS are given in this document. This document is directed to help programmers that begin to work with the system, to write application programs. Also, a list of existing constants (taken from the Assembler's TVDEF table) are given to help the programmer save core space.

MACRO INSTRUCTIONS CLASSIFICATION

Relocatable Pointers - RELPTR

Increment Pointers - INCRCP

Single Character Transfer - GETCHR, PUTCHR

Character Strings - CPRSET, LSTCHR, NEWSTR, TEMSTR, TOSTR

Identification Number - IDTOAD, ADTOID, IDTOCP

Teletype - TYPEIN, TYPOUT, TPTEXT, TPCRLF

Miscellaneous - WAIT, DATIME

RELOCATABLE POINTER

RELPTR (Relocate Pointer). Store in .RPn the contents of the A-register. The contents of .RPn will be adjusted by the system when a core shift occurs.

Calling Sequence

```
RELPTR          .RPn
    normal return
```

Remarks

The core space taken by a segment, that is not being used, can be released by the user with the two instructions given below.

```
CRA
RELPTR          .RPn
```

When a core shift occurs, the system will use the released space for new segments.

INCREMENT POINTER

INCRCP (Increment Character Pointer) Increment a character pointer, given as an argument, by the contents of the A-register.

Calling Sequence:

```
LDA       (increment used)
INCRCP    .Tn
normal return
```

SINGLE CHARACTER TRANSFER

GETCHR (Get character). Get a byte from a segment file and return it, right adjusted, in the A-register.

Calling Sequence

```
GETCHR          .Tn
error return        (ran out of characters)
normal return
```

Remarks

The character count stored in .Tn is incremented automatically after execution.

PUTCHR (Put character) Store the byte in the A-register in a buffer created by NEWSTR, TEMSTR, etc.

Calling Sequence

```
PUTCHR          .Tn
    Normal return
```

Remarks

There is no error return because the system creates a new segment when one is filled up without any action by the user. Also, the relative character counter is incremented automatically after execution.

CHARACTER STRINGS

CPRSET (Character Pointer Reset) Reset character pointer to the beginning of string.

Calling Sequences:

1. CPRSET A, B (where A = segment name and B = RA

normal return

2. CPRSET ,.Tn (contains a pointer to .RPn and the character pointer count)

normal return

Remarks:

A comma has to preceed the RA even if there is not a segment name. Upon execution of 2, .Tn will contain a pointer to .RPn and "0" for a character pointer. Also, .RPn will have the absolute starting address of the first word of the first segment of a chain.

LSTCHR (Last character) Define the last character of an existing string.

Calling Sequence

```
CALL    LSTCH
ADDR    .Tn
```

normal return

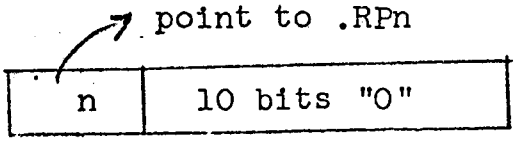
Remarks:

It truncates an existing character string, deallocates succeeding segments of the string, changes the forward pointer of the segment to zero, and sets up a last character indication in the final segment.

NEWSTR (New String) Creates the first data segment of a chain of segments. The high order bit of the segment header is ON. This means that the segment will be written on disc when the segment is released.

Calling Sequence

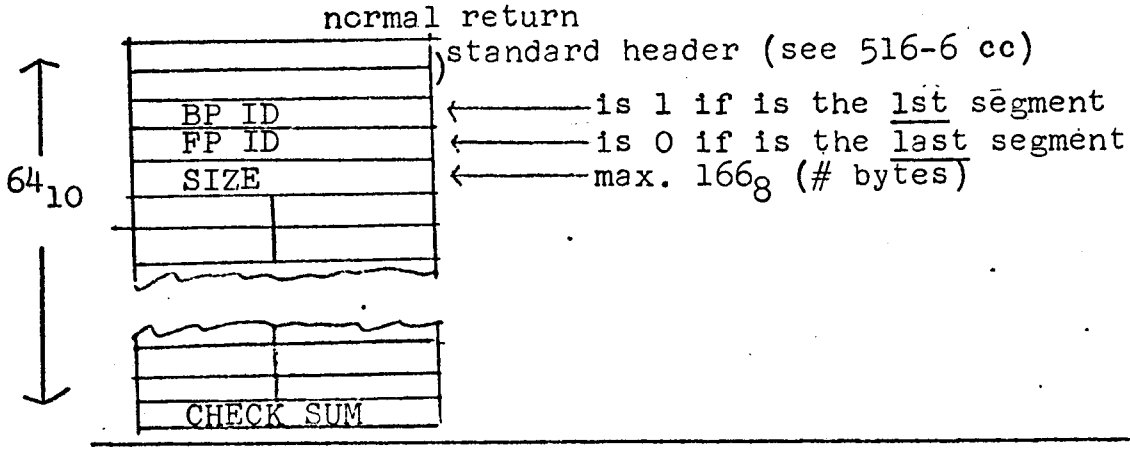
```
LDA     .CPRPn
STA     .Tn
NEWSTR
  normal return
```



TEMSTR (Temporary String) - Same as NEWSTR but the high order bit of the segment header is OFF and the segment is designated a temporary data segment. Therefore, the segments will not be saved on disc.

Calling Sequence:

```
LDA     .CPRPn
STA     .Tn
TEMSTR
  normal return
```



TOSTR (Throw out String) Delete String.

Calling Sequence

```
LDA     .Tn   - (see NEWSTR)
TOSTR
```

IDENTIFICATION NUMBER

IDTOAD (Identification Number to Address) - Given the ID # of a segment find its starting address.

Calling Sequence

```
        LDA     (ID - number of segment)
        IDTOAD
             normal return
```

Remarks:

The absolute starting address of the segment will be returned in the A-register. It is recommended that the user store this address in an .RPn pointer using the command RELPTR .RPn. An unallocated ID will drive the thread into the octal package.

Example: Store in .RPn the starting address of the segment whose ID number is stored in location PASS.

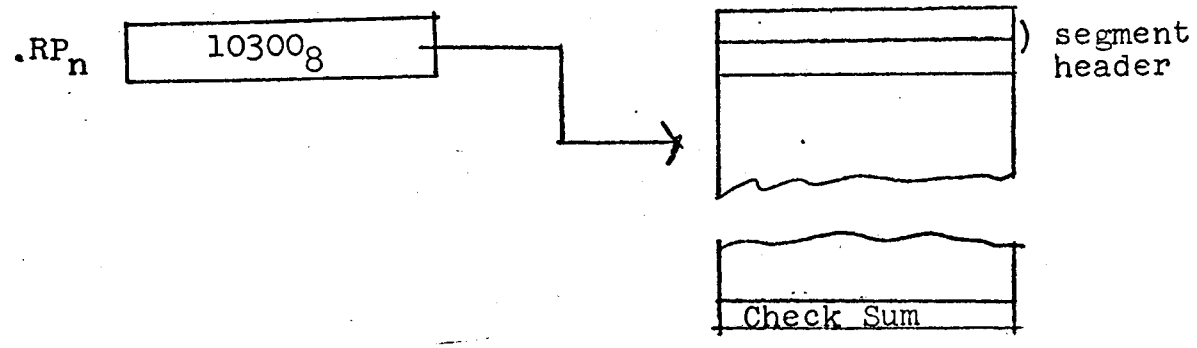
```
        LDA     PASS
        IDTOAD
        RELPTR  .RPn
          .
          .
          .
          .
PASS    ØCT     0
```

ADTOID (Address to Identification number) - Convert a segment absolute address stored in the A-register to its equivalent relative address inside the segment. The relative address is returned in the B-register and the ID number of the segment is returned in the A-register.

Calling Sequence

```
LDA   (relocatable pointer, i.e., .RPn)
ADTOID
  normal return
```

Example: Pointer .RPn contains an arbitrary address inside a segment. Find its ID number and its relative address. Store them in buffer words labeled ID and RA respectively. Assume that the segment starts at 10000_8. An error will drive the thread into the octal package.



Solution

```
LDA    .RPn
ADTOID
STA    ID
IAB
STA    RA
  .
  .
  .
  .
ID : OCT   0
```

IDTOCP (ID number to Character Pointer). Store the absolute starting address of a segment in an .RPn pointer.

Calling Sequence

```
        LDA     .CPRn
        STA     .Tn
        LDA     (ID number)
        IDTOCP  .Tn
          normal return
```

RPn	10 bits of "0"

Remarks: This macro is useful for character manipulation.

Example: Given the ID number of a segment containing ASCII characters store the first one of them in the A-register.

```
        LDA     .CPRPn
        STA     .Tn
        LDA     IDPASS
        IDTOCP  .Tn
        GETCHR  .Tn
          .
          .
          .
          .
IDPASS  OCT     (octal number)
```

.RPn	10 bits of "0"

TELETYPE

TYPEIN (Type In) Input a string of characters through the TTY.

Calling Sequence:

```
LDA     (character pointer, to start input buffer,
         i.e., LDA .Tn)
STA     .IOCP  (predefined address on base sector)
LDA     (termination character)
STA     .IØTCH  (predefined address on base sector)
TYPEIN
```

Normal return

TYPOUT (Type Out) Output a string of characters to the TTY.

Calling Sequence: Same as above but use TYPOUT instead of TYPEIN.

Remarks:

Returns updated character pointer or an indication of output interruption in the A-register (-1).

TPTEXT (Type Text) Output on the TTY, a text stored in a program segment.

Calling Sequence:

```
TPTEXT    (symbolic address of the text to be
           printed)
     normal return
```

Remarks: The last character on the text should be a reverse slash, \ .

Example:

```
TPTEXT    LOLA
 normal return
   .
   .
   .
   .
LOLA:ASCII (
HOLIDAY.\ )
```

TPCRLF (Type Carriage Return-Line Feed)

Calling Sequence:

```
TPCRLF
  (normal return)
```

MISCELLANEOUS MACROS

WAIT — Roadblock this thread for n seconds. (Turn control over to another user for n seconds).

Calling Sequence:

```
JST     .WAIT,*
DEC  n  (time in seconds after which control
         is returned)
```

DATIME — Type out the date and time on the TTY.

Calling Sequence:

```
CALL DATIME
  (normal return)
```

CONSTANTS

TTY. Characters

Symbol	Meaning	Equivalent	Number
.CR	carriage-ret	OCT	15
.SP	blank	OCT	40
.COMMA	,	OCT	54
.FS	/	OCT	57
.ASDGT	ASC 260	OCT	60
.BS	\	OCT	134
.LA	←	OCT	137
.SCR	carr. ret w/parity	OCT	215
.RBOUT	rub out	OCT	377
.CRLF	carr. ret line feed	OCT	6412

Symbol	Equivalent	Number
.M8	DEC	-8
.M4	DEC	-4
.M3	DEC	-3
.M2	DEC	-2
.M1	DEC	-1
.P0	DEC	0
.P1	DEC	1
.P2	DEC	2
.P3	DEC	3
.P4	DEC	4
.P5	DEC	5
.P6	DEC	6
.P7	DEC	7
.P8	DEC	8
.P9	DEC	9
.P10	DEC	10

The constants which start with .X are the hexadecimal equivalents of the last 3 or 4 characters of its name. A = 10, B = 11, C = 12, D = 13, E = 14, F = 15, i.e, OCT 77777 —— binary 111 111 111 111 111 — .X7FFF
7 15 15 15 15

.X200	OCT	1000
.X1000 , .CPRP1	OCT	10000
.X2000 , .CPRP2	OCT	20000
.X3000 , .CPRP3	OCT	30000
.X4000 , .CPRP4	OCT	40000

Symbol	Equivalent Number	
.X4001	OCT	40001
.X5000 , .CPRP5	OCT	50000
.X5800	OCT	54000
.X6000 , .CPRP6	OCT	60000
.X7000 , .CPRP7	OCT	70000
.X7FFF	OCT	77777
.X8000	OCT	100000
.X9000	OCT	110000
.XA000	OCT	120000
.XB000	OCT	130000
.XC000	OCT	140000
.XD000	OCT	150000
.XE000	OCT	160000
.XF000	OCT	170000
.XF800	OCT	174000
.XFC00	OCT	176000