516-41
ADH,CC
9/28/70

516 SEGMENT ASSEMBLER

A DDP-516 segmented program has been written to assemble DDP-516 segmented programs. The symbolic input format differs from the GMAP segment assembler (516-14), although the differences are small enough to permit straightforward (but laborious) conversion between the two formats. A conversion program, described below, is available to help the user convert the GMAP to 516 format. The assembler output is a binary object file and an optional listing file. The listing file is similar to the source file except that the octal location counter value is inserted in front of each line, showing counter content before that line has been processed.

The assembler is called from the Executive, in response to the query

PROGRAM?

by the response

ASSEMB,source[,option]

where the second argument is the name of the source file, and the optional third argument specifies an option. In all cases, a binary object file is produced. The options are as follows:

null [=blank]	produce a listing file
n	no listing file
l	produce listing file, load object file
p	produce listing file, send files to GE to print listing file and punch object file.
a	produce listing file, send files to GE, and load object file (combine options l and p).

In order to send files to the GE 635, the user must have a file named IDENT in his directory, and the IDENT file must contain the user's GE IDENT card.

The assembler names the binary object file by suffixing *B to the source file name; it names the listing file by suffixing *L to the source file name. Previous data in these output files, if any, is overwritten. If these files do not exist, they are created and inserted in the user's directory.

If an assembly has been run without loading the object file (null, n or p options), the object file may be loaded by SEGFIL without a reassembly, by the call from the Executive

SEGFILE,source*B

where source*B is the name of the object file. It should be noted that the object file format is the same as if the binary deck had been loaded from the card reader by the EDITOR. Also, the binary deck format is identical to that produced by the GMAP assembler, including the fact that the segment name is punched in columns 73-78, and a sequence number is punched in columns 79-80.

Free-field format

The source program is assumed to begin at the start of the source file, and end with an END card (card=line). Each card is written in a free-field format, as follows:

$label_1$:...$label_n$: op variable field\comments

where blanks may be freely used or omitted between fields and within expressions. The label field may be omitted or may contain any number of labels, where each label is followed by a colon. The blank or null label is reserved as a special pseudo-operation code, which causes the following data to be interpreted as an instruction definition. The operation code is normally terminated by a blank, but this is optional if the first character of the variable field is a break, e.g., comma or left parenthesis. The blank operation code is treated as a null; hence, blank cards are acceptable for formatting purposes in the source deck. The variable field is similar to the GMAP variable field, except that blanks may be interspersed between arguments or within expressions, as required. The variable field ends (except for the ASCII statement) with the end of the line or with a left slant. Comments may be placed between the left slant and the end of that line.

Symbols

Symbols may be composed of up to 6 contiguous characters from the set of alphabetics (lower case=upper case), numerals, and period. Asterisk is also a legal symbol character, but its use is reserved exclusively for system symbols; in particular, * is the symbol for the current location counter value. The null or blank symbol is equivalent to absolute 0 in value.

A symbol has a 16-bit value. A symbol is evaluated by first attempting to interpret it as an integer, according to the appropriate mode (octal, decimal, or hexadecimal, depending on context). If this fails, the symbol value is looked up in the symbol table. If it is not in the table, it is looked up in the table of System Symbols (TVNAME's). Hence the GMAP TVNAME pseudo-operation for naming System Symbols or defining equivalences to them is not needed and not available.

Segment names are recognized as such by context. Segment names must not be used as ordinary symbols or appear in expressions. It is likely that such a conflicting use will not be detected by the assembler.

Expressions

Expressions are alternating sequences of symbols and the operators + or -. If an expression begins or ends with an operator, there is assumed to be a leading or trailing null symbol, respectively. The assembler does not specifically check for relocation errors, so the user must exercise appropriate caution.

Machine operations

The operation codes are the same as for the GMAP assembler, except that mnemonics with prefixed dots are not defined (hence, use LDX, not .LDX). The variable field may contain an expression; any integers are interpreted in decimal mode. In the case of a memory-reference instruction, there may be a second subfield separated from the address field by a comma; if this second field contains *, indirect addressing mode is set. The tag bit is set implicity by the address: a relocatable address has a tag bit, an absolute address does not. It is easy to convert between absolute and relocatable addresses by using the predefined symbol .RZERO (relocatable zero): Add .RZERO to an absolute address to make it relocatable; subtract .RZERO from a relocatable address to make it absolute. (Indirect addressing may also be implicitly specified in the address field by the expression value.)

Address data

The ADDR operation code is used to specify a full-word address. The address and modifier fields are treated exactly as for memory-reference instructions, except of course that the address range is not limited to 512 words.

The VADDR operation may be used to generate a two-word virtual address. The first subfield contains the segment name; if this field is blank or null, segment zero is used (an absolute address). The second subfield contains an expression (decimal mode) for the ra; if this subfield is absent, the ra is zero.

The .VADR1 operation generates an id only, i.e., it is like the "first half" of VADDR. The segment name is given in the variable field.

External Linkage

The CALL pseudo-operation provides subroutine linkage to an external segment. CALL takes one argument, in the same format as VADDR, that is, a segment name in the first subfield, an ra in the second subfield.

The GOTO pseudo-operation provides a transfer linkage to an external segment. Like CALL, it takes a single, two-component argument.

Symbol-Defining Pseudo-Operations

It has already been explained how symbols may be defined by appearing as labels. A trio of pseudo-operations, SET, BØØL, and SETX, have been provided to permit the user to equate a symbol to an expression. The symbol to be defined (or redefined) is given as the first argument; the expression is given as the second argument. Any symbols appearing in the expression must have been previously defined. The particular pseudo-operation governs the interpretation of integers in the expression: SET-decimal, BOOL-octal, SETX-hexadecimal.

Two symbols have been predefined. Asterisk (*) represents the current value of the location counter. .RZERO is relocatable zero, i.e., 40000_8 (the index bit).

Operation-Defining Pseudo Operations

As mentioned previously, an empty label (an isolated colon) is used to define operation codes. The first argument is the operation symbol, the second argument is the operation value,

and the third argument is the operation type. Integers are interpreted as hexadecimal. It is not recommended that users define operations this way, however, because the interpretation of value and type codes is subject to change.

A user operation, OPSYN, is provided to permit the user to define (or redefine) an operation as equivalent to another operation. The first argument is the operation to be defined, the second argument is the operation to which it is equated.

Data-Generating Pseudo-Operations

The pseudo-operations OCT, DEC, and HEX are available so that the user may generate octal, decimal, or hexademical integer data words, respectively. Each pseudo-operation generates as many data words as there are arguments (i.e., one plus the number of commas). The integers may be signed; if negative, they are represented in two's-complement form.

An ASCII pseudo-operation is available for generating text data. The argument string begins with a left parenthesis and ends with a right parenthesis. If the number of characters is odd, the last word contains a rubout (377$_8$) character in the right half. The left slant (\) character generates a null (0) character. Carriage return is legal; hence, ASCII may have a multi-line format. (Note: insertion of carriage return causes a line-feed to automatically be generated.)

Because it is impossible to generate certain characters with ASCII, such as \ or), a character or half-word trio of pseudo-operations has been provided to permit numerical definition of characters. The first argument defines the left (most significant) half-word, the second argument defines the right (least significant) half-word. Integers are interpreted according to the code: OCHAR-octal, DCHAR-decimal, XCHAR-hexadecimal.

Storage Allocation

One storage allocation pseudo-operation has been defined, BSS. The location counter is advanced by the value of the expression in the variable field; integers are interpreted in decimal.

End and Segment Name

The END pseudo-operation signals the end of the source deck. It also supplies the segment name, in the variable field.

Conversion of GMAP Format Programs

A program called CVGEAF has been written to convert GE to 516 symbolic format. The source file is named as an argument in the call, i.e.,

```
CVGEAF,source
```

The conversion consists in placing a colon after each label and inserting a left slant (\) in column 26 if column 26 is blank. The conversion is effected by changing the source file, not by providing an altered copy of it. A left slant also replaces an asterisk in column 1.

Errors

Errors are flagged and typed out to the user during the second (output) pass, but not during the first (definition) pass. The format is an error character followed by the line number (in decimal) at which the error occurred. The error characters are somewhat ambiguous in their meanings, but in general, they have the following significance:

F	End of file (e.g., no END card).
U	Undefined symbol or operator, usually in an expression.
A	Address error, e.g., machine instruction address greater than 511.
M	Multiple definition; a location symbol is being redefined, with a new definition different from the old definition. The new definition is allowed to supersede the old one. This may also indicate a phase error, that is, a location symbol has a different value in pass 2 than it had in pass 1.
O	Operation code error; operation code is undefined or has an impossible definition. In any case, the operation is treated as null (the location counter is not bumped).