# AP1 - AP2 PROGRAMMING

## I. Symbolic Coding.

Symbolic coding for the Rice Computer is accomplished through the use of either of two assembly programs: AP1, designed for independent use, or AP2, designed for use within Genie compiler language. The two assembly systems are very similar; minor differences between them will be noted below. The major distinction concerns octal and decimal numerals. In AP1, all numerical constants are assumed to be octal unless immediately preceded by the special symbol "d", meaning decimal. In AP2, all numerical constants are assumed to be decimal, except when octal form is indicated by a plus sign immediately preceding the octal number. In the following discussions, M stands for the final address formed in the last 15 bits of I (the instruction register) after all indirect addressing and B-modification has taken place; and if Q is any machine location, then (Q) stands for the contents of location Q.

### I.1. Instruction form.

The general form of a single AP1 or AP2 instruction is

| LOCN | SETU | OPN | ADDR+MOD, AUX |
|---|---|---|---|
| cr | 1st tab | 2nd tab | 3rd tab |

where "tab" denotes "tabulate", and "cr" denotes "carriage return". This corresponds to the absolute form of the instruction, as explained in the Rice Computer Manual. LOCN gives the symbolic form (if any) of the location of the instruction. SETU corresponds to Field 1: bring a "fast" register to U; then inflect (U). OPN corresponds to a Field 2 operation chosen from one of six classes (see section II). AUX corresponds to Field 3: alter a B-register, send (U) or (R) to a "fast" register, send the M portion of I to a B-register, or clear R. ADDR+MOD corresponds to Field 4: compute the final address M, sending M to the last 15 bits of I; load S with M or (M); then inflect (S). Any field except LOCN may contain absolute octal codes, which will be placed in the corresponding positions of the machine instruction, ignoring any bits which overflow to the left. (An exception to this rule is discussed under ADDR+MOD symbols in Section I.3.) LOCN may also be absolute in certain cases (see the ORG pseudo-order in Section III.4). AUX and the comma preceding it may be omitted.

### I.2. Symbolic language.

Precise definitions of the allowed symbols are as follows:

Type I: Special symbolic addresses. By convention we

recognize the following symbols for "fast" addresses: Z, U, R, S, T4, T5, T6, T7 (A-series); and CC, B1, B2, B3, B4, B5, B6, PF (B-series). These may appear in SETU, ADDR+MOD, and AUX fields. Use of the above symbols, and of X (for the increment register) and I (for the instruction register) should be restricted to the special significance associated with the Rice Computer.

Type II: Special characters. *, a (AP1) or # (AP2), d (AP1), +, -, |, →, (, ), "tab", "cr", and , (comma).

Type III: Mnemonic operation codes as listed in Section II; AP1 pseudo-operation codes as discussed in Section III.4; and symbols previously defined as operation codes by means of a LET (AP2) or EQU (AP1) instruction (see the EQU pseudo-order in Section III.4).

Type IV: General storage addresses. In AP2, any private name formed by the rules described in the Notes on Genie. In AP1, any upper case Roman letter, which may be followed by upper case Roman letters, or numerals; any number of characters are permissible, but a maximum of six will be retained by AP1. (A letter may not follow a numeral; BA3 is permissible, but not B3A.) Examples: B, M3, COMM, ZETA2. These symbols may appear only in the LOCN or ADDR fields.

I.3. Contents of instruction.

Each field of the symbolic instruction has a well-defined form, and if this is not recognized by the assembly system, a note is made on the printed listing of the program. The acceptable contents of each field are as follows:

LOCN. May be blank or absolute or symbolic. Absolute LOCN fields are permitted only when an AP1 program is being assembled in absolute form (see the ORG pseudo-order in section III.4). Symbolic LOCN fields may consist of any Type IV symbol, and may appear in conjunction with a relative numerical part, as "LOOP+1", "EXIT-3".

SETU. May be blank or absolute or F, where F is an A- or B-series symbolic address, or any of the forms -F, |F|, or -|F|. If SETU is blank, the symbol "U" is understood and the octal equivalent 01 is inserted into the machine instruction. The symbols |Z| and -|Z| have special meanings: |Z| sets U to the numerical integer +1; -|Z| sets U to the numerical integer -1. Note that Z sets U to all zeros; -Z sets U exponent to zero and U mantissa to minus zero, or all ones.
Examples: B1 |T4| -PF -|R|

OPN. May be any absolute octal code or Type III symbol. In the case of conditional transfers, a symbolic operation has

the form IF(CCC)TTT where CCC represents test conditions and TTT is a mnemonic for a transfer order. Other symbolic operation codes consist of one or more 3-letter mnemonics. Special symbols such as →, +, -, ",", and +i (where i is an integer) are sometimes permitted (see the listing of OPN codes in Section II).

AUX. May be blank, absolute, or one of the forms U→F, R→F, I→Bi, Bi+1, Bi-1, or Bi+X, where Bi stands for one of the Type I B-series symbols, F is any Type I A- or B-series symbol, I refers to the last 15 bits of the instruction register, and X refers to the increment register. Note that R→Z causes R to be cleared to zero.

Examples: U→T4 R→PF I→B1 B2+1 B3-1 B4+X

ADDR+MOD. ADDR may be blank or absolute or symbolic, or the ADDR+MOD field may consist of an octal or decimal number to be used as an operand. MOD is either blank or one or more of the Type I B-series symbols, connected to ADDR by + signs. Special inflections control the IM and IA bits as follows: IM bit 1 is set to 1 whenever the symbol "a" (AP1) or "#" (AP2) appears, or whenever certain OPN mnemonics are used (see the listing of OPN codes in Section II). IM bits 2 and 3 are controlled by the special forms -Q, |Q|, and -|Q|, where Q is an allowed ADDR+MOD symbol. The IA bit is set to 1 whenever the symbol "*" appears in this field.

If ADDR is symbolic, any Type I A-series symbol, X (the increment register), or any Type IV symbol is acceptable. As in LOCN, this field may contain a relative part consisting of an integer preceded by a + or - sign.

If ADDR is absolute, any octal address in the range 0-20007, or any decimal address in the range 0-8191, is acceptable. Likewise, any octal integer of not more than 5 digits, or any decimal integer of absolute value not larger than 32,767, is permissible. Any octal or decimal integer above these limits, or any floating point decimal number (see the DEC pseudo-order in Section III.4), is treated as an operand; storage space is reserved for it at the end of the program. In this case, the MOD, "a" or "#", and "*" symbols must be omitted, but the other IM inflections may be present.

All characters appearing within parentheses in this field are treated as the Z character, so that an address which is modified by the program may be conveniently noted. For example, (FWA)+B1+B2 is treated as Z+B1+B2. If a symbol appears in ADDR but never in LOCN, a blank location will be reserved at the end of the program; this is true even when such a symbol appears only within parentheses. ADDR and MOD should not both be blank; the Z character may always be used to produce a zero field.

Examples of equivalent AP1 and AP2 ADDR+MOD fields are:

| AP1 | AP2 |
|---|---|
| COMM+10 or COMM+d8 | COMM+8 or COMM+#10 |
| -\|A+B1-d12\| or -\|A+B1-14\| | -\|A+B1-12\| or -\|A+B1-+14\| |
| a*ZETA | #*ZETA |
| d48 | 48 |
| -ad122+B1 | -#122+B1 |
| B4+B5 | B4+B5 |
| 00500 | +00500 |
| d2.009027 | 2.009027 |
| 777700000 | +777700000 |

The only field which may be continued onto another line is "ADDR+MOD, AUX". This is achieved by punching a "cr" followed by three "tab" characters, so that continuation lines will follow under "ADDR+MOD, AUX". Any number of "cr" characters may be punched to help separate code sequences on the printed page; they will appear in symbolic listings, but will be ignored during assembly. Comments punched with a 7th hole in the tape may be included for the guidance of the coder; they are not read by the computer and hence will not appear on machine listings of the program.

## II. Mnemonic operation codes.

The most common Field 2 operations have been assigned symbolic equivalents in AP1 and AP2 for convenience in coding. These Type III symbols may not be used for any other purpose. Other Field 2 operations may be assigned symbolic equivalents by a LET (AP2) or EQU (AP1) statement (see the EQU pseudo-order in Section III.4); such symbols are then treated as Type III symbols throughout the program in which they have been defined. In the list below, the symbols are followed by their octal equivalents and a brief explanation of their meanings; the indication "a,#" means that the operation symbol automatically causes 1M bit 1 to be set to 1, since the operation indicated deals with M rather than with (S).

### II.1. Class 0.

The four unconditional transfers are represented by:

| | | | |
|---|---|---|---|
| a,# | HTR | 00000 | Halt and transfer. Halt, setting CC to M when the Continue button is pressed. |
| a,# | TRA | 01000 | Transfer. Set CC to M. |
| | SKP | 02000 | Skip. Subtract (S) from (U); then increment CC by 1, skipping the next order. |
| | JMP | 03000 | Jump. Subtract (S) from (U); then increment CC by (X), the increment register. |

Conditional transfers have the form IF(CCC)TTT where TTT is one of the above transfer mnemonics, and CCC represent one, two, or three test conditions joined by + or X signs. Use of the + sign indicates that the transfer specified is to occur if any of the conditions listed is satisfied; use of the X sign, that the transfer occurs only when all of the conditions listed are satisfied simultaneously. A single order may not contain both + and X signs. One condition from each of the first three groups may be specified; or a Group IV mnemonic may be combined with a Group III test as noted below.

Group I

| | | | |
|---|---|---|---|
| | MOV | 00200 | Mantissa overflow. Is Indicator Light #4 on? |
| | EOV | 00300 | Exponent overflow. Is Indicator Light #5 on? |
| | NMO | 00600 | No mantissa overflow. Is Indicator Light #4 off? |
| | NEO | 00700 | No exponent overflow. Is Indicator Light #5 off? |

Group II

| | | | |
|---|---|---|---|
| | ZER | 00010 | Zero. Is (U) mantissa all 1's or all 0's? |
| | EVN | 00020 | Even. Is bit 54 of U equal to zero? |
| a,# | SLN | 00030 | Sense light on. Are all the sense lights corresponding to 1's in M on? |
| | NUL | 00040 | Null. Are all 54 bits of U zero? |
| | NZE | 00050 | Non-zero. Is (U) mantissa different from zero? |
| | ODD | 00060 | Odd. Is bit 54 of U equal to 1? |
| a,# | SLF | 00070 | Sense light off. Are all the sense lights corresponding to 1's in M off? |

If the SLN or SLF test is used with a SKP or JMP order, no subtraction takes place. If the NUL test is used with a SKP or JMP order, a logical comparison is made as follows: wherever a bit of R is equal to zero, the bits in corresponding positions of U and S are compared. If (U) is identical with (S) in each of these positions, the resulting (U) is null and the NUL portion of the test is satisfied.

Group III

| | | |
|---|---|---|
| TG1 | 00001 | Tag 1. Is Indicator Light #1 on? |
| TG2 | 00002 | Tag 2. Is Indicator Light #2 on? |
| TG3 | 00003 | Tag 3. Is Indicator Light #3 on? |
| NTG | 00004 | No tag. Are Indicator Lights #1, #2, #3 all off? |
| NT1 | 00005 | No tag 1. Is Indicator Light #1 off? |
| NT2 | 00006 | No tag 2. Is Indicator Light #2 off? |
| NT3 | 00007 | No tag 3. Is Indicator Light #3 off? |

Note that indicator lights are turned off when tested; sense lights are not altered when tested.

Group IV

| | | |
|---|---|---|
| POS | 00110 | Positive. Is (U) mantissa greater than or equal to zero? |
| NEG | 00510 | Negative. Is (U) mantissa less than or equal to zero? |

A + sign must be used when combining either of these mnemonics with a Group III test.

| | | |
|---|---|---|
| PNZ | 04150 | Positive non-zero. Is (U) mantissa greater than zero? |
| NNZ | 04550 | Negative non-zero. Is (U) mantissa less than zero? |

A X sign must be used when combining either of these mnemonics with a Group III test.

II.2. Class 1.

Any Class 1 mnemonic may be followed by → or +1, to cause storing of the final (U) in the location addressed by M; or by +3, storing (U) at location M+(B6). Any floating point mnemonic may be followed by +1j (j=0, 1, or 3), causing (U) to be rounded (before storing); or by +6j, suppressing normalization of the result in U; or by +7j, to obtain rounding without normalization. In addition, SUB may be followed by +400j, and FSB may be followed by +40ij (i=0, 1, 6, or 7; j=0, 1, or 3) to interchange (U) and (S) before subtracting. The Class 1 mnemonics are as follows:

| | | |
|---|---|---|
| ADD | 10000 | Add. (U)+(S)→U. |
| SUB | 10100 | Subtract. (U)-(S)→U. |
| MPY | 10200 | Multiply. (U)×(S)→U,R (double length). |
| DIV | 10300 | Divide. Double length (U,R)÷(S)→U, remainder →R. |
| VID | 16300 | Reverse divide. (S)÷(U)→U, remainder →R. |
| IDV | 13300 | Integer divide. (U)÷(S)→U, remainder →R. |
| VDI | 17300 | Reverse integer divide. (S)÷(U)→U, remainder →R. |
| FAD | 10400 | Floating add. (U)+(S)→U. |
| FSB | 10500 | Floating subtract. (U)-(S)→U. |
| FMP | 10600 | Floating multiply. (U)×(S)→U. |
| FDV | 10700 | Floating divide. (U)÷(S)→U, remainder →R. |
| VDF | 16700 | Reverse floating divide. (S)÷(U)→U, remainder →R. |

II.3. Class 2.

Any Group I or Group II mnemonic may be followed by a

comma and any Group III mnemonic. In addition, any Group I or Group III mnemonic may be followed by → or +1, storing (U) with (ATR) at location M; or any Group I, II, or III mnemonic may be followed by +3, storing (U) with (ATR) at location M+(B6). Note that all Group I and Group II mnemonics clear (ATR) unless followed by a Group III mnemonic. The Class 2 mnemonics are as follows:

| | | | |
|---|---|---|---|
| Group I | | | |
| | CLA | 21700 | Clear and add. Bring (S) to U. |
| | BEU | 21000 | Bring exponent to U. Exponent portion of (S) replaces exponent portion of (U). |
| | BMU | 20700 | Bring mantissa to U. Mantissa portion of (S) replaces mantissa portion of (U). |
| | BLU | 21400 | Bring left half to U. Left half of (S) replaces left half of (U). |
| | BRU | 20300 | Bring right half to U. Right half of (S) replaces right half of (U). |
| | BAU | 20100 | Bring address to U. Address portion of (S) replaces address portion of (U). |
| BEU, | BRU | 21300 | Bring exponent and right half to U. |
| BEU, | BAU | 21100 | Bring exponent and address to U. |
| BLU, | BAU | 21500 | Bring left half and address to U. |
| Group II | | | |
| | RPE | 20701 | Replace exponent. Exponent portion of (U) replaces exponent portion of word at location M. |
| | RPM | 21001 | Replace mantissa. Mantissa portion of (U) replaces mantissa portion of word at location M. |
| | RPL | 20301 | Replace left half. Left half of (U) replaces left half of word at location M. |
| | RPR | 21401 | Replace right half. Right half of (U) replaces right half of word at location M. |
| | RPA | 21601 | Replace address. Address portion of (U) replaces address portion of word at location M. |
| a,# | STO | 20001 | Store. Store (U) at location M. |

Note: "Replace" mnemonics may not be combined with each other.

| | | | |
|---|---|---|---|
| Group III | | | |
| | ST1 | 20010 | Set Tag 1. Set ATR to 1. |
| | ST2 | 20020 | Set Tag 2. Set ATR to 2. |
| | ST3 | 20030 | Set Tag 3. Set ATR to 3. |
| | WTG | 20040 | With Tag. Do not change ATR. |
| Group IV | | | |
| | NOP | 20040 | No operation. Do not alter (U) or (ATR). |
| | FST | 20041 | Fetch and store. Bring contents of location M to S; then store (U) with (ATR) at location M. |

II.4. Class 4.

The Class 4 mnemonics are as follows:

| | | | |
|---|---|---|---|
| a,# | TSR | 40000 | Transfer to subroutine. Set PF to (CC); then set CC to M. |
| a,# | SBi | 4000i | Set Bi. Set Bi to M, for i=1, 2, ..., 6. |
| a,# | SPF | 40007 | Set PF. Set PF to M. |
| a,# | ACC | 41000 | Add to CC. (CC)+M→CC. |
| a,# | ABi | 4100i | Add to Bi. (Bi)+M→Bi, for i=1, 2, ..., 6. |
| a,# | APF | 41007 | Add to PF. (PF)+M→PF. |
| | ERM | 00020 | Enter repeat mode. Turn on mode light #2. |

The ERM mnemonic is meaningful only when joined by a comma to one of the above Class 4 mnemonics.

| | | | |
|---|---|---|---|
| a,# | SLN | 42000 | Sense light on. Turn on sense lights corresponding to 1's in M. |
| a,# | ILN | 42001 | Indicator light on. Turn on indicator lights corresponding to 1's in M. |
| a,# | MLN | 42002 | Mode light on. Turn on mode lights corresponding to 1's in M. |
| a,# | TLN | 42003 | Trap light on. Turn on trapping lights corresponding to 1's in M. |
| a,# | SLF | 42004 | Sense light off. Turn off sense lights corresponding to 1's in M. |
| a,# | ILF | 42005 | Indicator light off. Turn off indicator lights corresponding to 1's in M. |
| a,# | MLF | 42006 | Mode light off. Turn off mode lights corresponding to 1's in M. |
| a,# | TLF | 42007 | Trap lights off. Turn off trapping lights corresponding to 1's in M. |

Note that lights corresponding to 0's in M are not affected by the above orders.

| | | | |
|---|---|---|---|
| a,# | DMR | 44000 | Double mantissa right. Arithmetic right shift of (U,R) mantissa M places as diagrammed in the Rice Computer Manual. |
| a,# | DML | 44010 | Double mantissa left. Arithmetic left shift of (U,R) mantissa M places as diagrammed in the Rice Computer Manual. |
| a,# | LUR | 45010 | Logical U right. Shift (U) right M places, shifting zeros into left end of U. |
| a,# | LUL | 45020 | Logical U left. Shift (U) left M places, shifting zeros into right end of U. |
| a,# | LRR | 45001 | Logical R right. Shift (R) right M places, shifting zeros into left end of R. |
| a,# | LRL | 45002 | Logical R left. Shift (R) left M places, shifting zeros into right end of R. |
| a,# | LRS | 45015 | Long right shift. Shift (U,R) right M places, shifting (U) into R and zeros into left end of U. |

| | | | |
|---|---|---|---|
| a,# | LLS | 45062 | Long left shift. Shift (U,R) left M places, shifting (R) into U and zeros into right end of R. |
| a,# | CRR | 45055 | Circle right. Shift (U,R) right M places, shifting (U) into R and right end of (R) into left end of U. |
| a,# | CRL | 45066 | Circle left. Shift (U,R) left M places, shifting (R) into U and left end of (U) into right end of R. |
| a,# | BCT | 46000 | Bit count. Clear U; shift R right M places; add each 1 which spills from R one at a time into U. |
| a,# | STX | 43005 | Set X. Set the increment register to M. |

II.5. Class 5.

Any Class 5 mnemonic may be followed by → or +1, to cause storing of the final (U) at location M; or by +3, storing (U) at location M+(B6). In addition, any Class 5 mnemonic may be preceded by a - sign, causing the final result in U to be complemented (before storing). The Class 5 mnemonics are as follows:

| | | |
|---|---|---|
| CPL | 50100 | Complement. Change all 1's in U to 0's and all 0's to 1's. |
| XUR | 54000 | Exchange (U) and (R). (U)→R as (R)→U. |
| LDR | 50400 | Load R. (S)→R without disturbing (U). |
| LTi | 504i0 | Load Ti. (S)→Ti without disturbing (U) or (R), for i=4, 5, 6, 7. |
| ORU | 50010 | Or to U. Logical or: for every bit position, a one in U or a one in S (or both) results in a one in that position of U. A zero in any bit position of both U and S results in a zero in that position of U. |
| AND | 50314 | And. Logical and: for every bit position, a one in U and a one in S results in a one in that position of U. A zero in any bit position of either U or S results in a zero in that position of U. |
| XTR | 50020 | Extract. Wherever a bit of R is equal to one, the bit in that position of S replaces the corresponding bit in U. Other bits of U are unchanged. |
| SYM<br>SYD | 53220<br>53220 | Symmetric difference. For every bit position, a one in U and a zero in S, or vice versa, results in a one in that position of U. Two zeros, or two ones, in corresponding bit positions of U and S result in a zero in that position of U. |
| SYS | 53120 | Symetric sum. For every bit position, a one in U and a zero in S, or vice versa, results in a zero in that position of U. Two zeros, or two ones, in corresponding bit positions of U and S result in a one in that position of U. |

II.6. Class 6.

Either "Read" mnemonic may be followed by → or +1, storing (U) at location M; or by +3, storing (U) at location M+(B6). For detailed explanations of reading, printing, and punching, see the Rice Computer Manual. The Class 6 mnemonics are as follows:

| | | | |
|---|---|---|---|
| a,# | RTR | 60000 | Read triads. Read 1 to 18 triads from paper tape into U. |
| a,# | RHX | 60100 | Read hexads. Read 1 to 9 hexads from paper tape into U. |
| | PHX | 60400 | Punch hexads. Punch 1 to 9 hexads from (S) onto paper tape. |
| | PH7 | 60500 | Punch hexads with 7th hole. Punch 1 to 9 hexads, each with a 7th hole, from (S) onto paper tape. |
| | PTR | 60600 | Punch triads. Punch 1 to 18 triads from (S) onto paper tape. |
| | TYP | 60700 | Type. Type (S) as 18 octal digits on console typewriter. |
| a,# | PRN | 61110 | Print numeric. Print, using first 32 characters of print wheel, from print matrix beginning at location M; space one line after printing. |
| a,# | PRA | 61210 | Print alphanumeric. Print as above, using all characters. |
| a,# | PRO | 61310 | Print octal. Print as above, using characters 0-7 only. |

III. Placer and assembly operations.

III.1. Genie Placer.

The typing of Genie tapes containing AP2 instructions, and the handling of such tapes by the Genie Placer system and the Genie compiler, are just as described in the Notes on Genie. AP2 instructions may appear anywhere between the SEQ and END of a Genie language program, and may be interspersed with other Genie commands. One word of caution; though each AP2 command produces only one machine instruction, a single Genie command may produce several machine instructions. Therefore relative addressing and skip and jump commands should be used only within a continuous set of AP2 instructions.

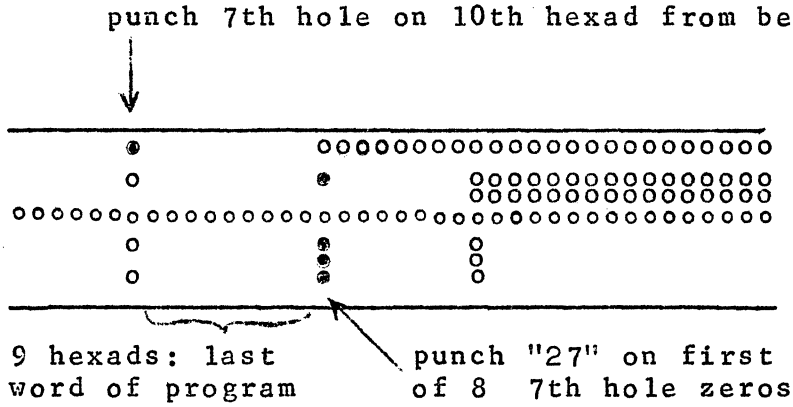
III.2. AP1 Placer.

The AP1 Placer system is located on the MT system magnetic tape at block 100.02. It differs from the Genie Placer system primarily in that a sixth sense light option exists, that of assembly. READ, EDIT, and PUNCH operations (sense lights 1, 2, and 3) are identical with those for Genie Placer. LIST (SL 4 on) prints only valid AP1 characters, replacing any other character with a backwards arrow. CHECK (SL 5 on) normally occurs next,

followed by ASSEMBLE (SL 6 on). CHECK is initially bypassed if the tape to be checked is not in the reader; Placer then proceeds to ASSEMBLE, if SL 6 is on, before making another attempt to CHECK. If the tape still is not in the reader, then the stop (I): 05 HTR CC occurs.

If only one sense light option is requested, the stop (I): 0i HTR CC (for SL i on) occurs; other sense lights may then be set for special forms of output. Pushing CONTINUE then causes the operation indicated to be carried out. For LIST (SL 4 on), setting SL 15 on when the stop (I): 04 HTR CC occurs causes double spacing on the listing. For ASSEMBLE (SL 6 on), the following options may be exercised by turning on the appropriate sense lights (in addition to lights 14 and 15 which are turned on automatically) when the stop (I): 06 HTR CC occurs:

SL 9 on: print with double (instead of single) spacing.
SL 11 on: do not punch assembled program.
SL 12 on: load for execution (permissible only for relativized programs; see the ORG pseudo-order below). Note that this option is useful only if the limited S-SPIREL loaded with Placer is sufficient for running the assembled program.
SL 13 on: punch self-loading tape (permissible only for absolute programs; see the ORG pseudo-order below). The paper tape produced by AP1 is punched in hexad form with check-sum, and is normally preceded by an M-SPIREL control word. By assembling an absolute program with SL 13 on, a tape will be produced which will load independently of M-SPIREL (by using the LOAD switch) after the following changes are made with the hand punch:




III.3. AP1 Assembly Output.

The printed output from AP1 assembly is interpreted as follows:

Error indications. An error indication is produced by apparent errors in syntax or sequencing. The type of error suspected is briefly indicated, followed by a line number from which it should be possible to detect the source of the error.

Note that line numbers refer to the partially assembled program which still contains pseudo-orders; location numbers refer to the final form of the program containing only valid machine instructions.

Symbol table. The table of symbols is printed out in seven columns giving information relevant to the symbols defined in the program:

(a) The relative position in the table.
(b) The symbol.
(c) A number (usually 0) which determines the type of object for which the symbol stands.
(d) The equivalent assigned to the symbol (5 octal digits).
(e) A number (usually 0) which determines whether or not an equivalent has been assigned. A number 2 indicates that a symbol remains unassigned and is a possible error in the final program.
(f) An 18 digit octal number. The first 5 digits indicate the line at which an equivalent was assigned.
(g) A number which indicates how (if at all) the equivalent was assigned:
0: by appearing in the LOCN field of an order.
1: by appearing in the LOCN field of an EQU pseudo-order in which the address was symbolic (see below).
2: by appearing in the LOCN field of an EQU pseudo-order in which the address was numeric (see below).

Assembled program listing. Five columns are printed, giving:
(a) The line count in octal form.
(b) The symbolic location (if any exists).
(c) The location count in octal form.
(d) The final instruction in octal form.
(e) The symbolic address (if any exists).

III.4. AP1 pseudo-orders.

Differences between line and location numbers, and skips in the sequence of line numbers, are caused by pseudo-orders as described below. These special instructions govern the process of AP1 assembly and facilitate the handling of blocks of various types of data within AP1 programs.

ORG and END. All programs to be assembled by AP1 must be preceded by an ORG (Origin) order and terminated by an END order. In each case the remaining fields in the symbolic instruction are interpreted in a special way. Each of these orders advances the line count by 1; the location count is not affected.

The function fo ORG is (a) to initialize the assembly process, (b) to identify the program which follows, (c) to determine whether it is to be assembled in relative or absolute final form, and (d) to give an approximate indication of its maximum size. The ORG order is preceded by a "cr" and an "uc" or "lc" punch (upper or

lower case); the order itself has the form 1 s ORG m, n where 1, s, m, and n stand for either absolute or blank fields (symbolic fields are not allowed). If 1 (LOCN) is not zero, it is taken to be an M-SPIREL index number for the relativized routine which follows; in this case m must be zero. The SETU field, s, is always blank or zero. If m (ADDR) is not zero, the following code is assembled in absolute form starting in location m; in this case 1 must be zero. (In a relativized program, if an order in location P refers in Field 4 to location Q, it is through a Control Counter reference of the form CC+(Q-P)-1. In an absolute program, the Field 4 reference is directly to location Q; and absolute LOCN fields, giving the location of an instruction in 5-digit octal form, are permitted.) If n is blank, it is assumed that the total number of lines in the following program is octal 200 or less (d 128). If n is not blank, its value is used in taking storage space for the program which follows. The value of n may be greater or less than 200, but not greater than 10,000 (d 4096).

The END order has the form b b END cr cr where b stands for a blank field. The END order must be immediately followed by two (or more) carriage returns.

EQU. The Equivalent order has the form 1 b EQU m where 1 (LOCN) is symbolic, b (SETU) is blank, and m (ADDR) is either absolute or a symbol whose value has previously been assigned, through its appearance in the LOCN field of another order. Then 1 is immediately assigned the value m. If m is a 5-digit octal OPN code, then the symbol 1 may appear in Field 2 of any order following the EQU order and will be replaced during assembly by the octal code for which it stands. This order advances the line count by 1; the location count is not affected.

BSS and BES. Either of these orders inserts a block of zero words into the body of the program. BSS (Block started by symbol) and BES (Block ended by symbol) have the form 1 b XXX m where 1 (LOCN) is blank or symbolic, b (SETU) is blank, and m (ADDR) is either absolute or a previously assigned symbol. The value of m determines the number of zero words to be inserted; if 1 is symbolic, it is assigned as if the LOCN field had been associated with the first (BSS) or last (BES) word in the block. Each of these orders advances the line count by m+1, and the location count by m.

BCD, FLX, REM. These orders deal with alphanumeric data and have the form 1 b XXX m where b (SETU) is always blank; in each case, the mnemonic must be followed by a "tab" character, and after that all characters (in the ADDR field m) are read and stored, 9 characters per word. Any occurrence of the "cr-tab-tab-tab" sequence is replaced by a "space", and the string of characters is terminated by a true carriage return, allowing more than one line of data to be given. For BCD (Binary Coded Decimal), each character is converted to a corresponding printer hexad; if 1 (LOCN) is symbolic, it is assigned as if associated with the first word stored. For

FLX (Flexowriter), all codes (including case shifts, etc.) are preserved without conversion; l may be symbolic as for BCD. For REM (Remarks), l must be blank; this order is used only to obtain printed comments in the program listing, and does not introduce any data into the final program. These orders advance the line counter by n+1, where n is the number of 9-character words stored; in addition, BCD and FLX advance the location counter by n.

DEC and OCT. The Decimal and Octal orders are used for inserting numerical data into the body of the program. They have the form l b XXX m where l (LOCN) is blank or symbolic, b (SETU) is blank, and m (ADDR) consists of a list of one or more octal or decimal numbers. If l is symbolic, it is assigned as if associated with the first number in the list. Each number must be separated from its successor by a comma, and each will occupy a separate word in the final program. Continuation lines should not be used; for long lists of numbers, several DEC or OCT pseudo-orders in succession may be used to produce a continuous block of data. An octal number consists of one to 18 octal digits. A decimal integer consists of one to 14 decimal digits; a floating point decimal number, of one to 14 significant figures and a decimal point. If the list m consists of n numbers, either of these orders will advance the line count by n+1 and the location count by n.

III.5. Placer and Genie requests.

A Placer or Genie request must be provided with each symbolic program to be processed; some of the options available may be omitted at the coder's discretion.

| Placer | Genie |
|---|---|
| ✓ Read Tape *253 | ✓ Read Tape PROG |
| ✓ Edit *253 E | ✓ Edit PROG E |
| ✓ Punch *253 S | ✓ Punch PROG S |
| ✓ List | ✓ List |
| ✓ Check | ✓ Check |
| ✓ Assemble *253 A | ********** |
| | ✓ Compile PROG A |

The original tape is read and a "tape image" is formed in the machine. The edit tape E is read and the tape image is edited. Then the edited program is punched and listed, and tape S is checked for agreement with the tape image. Placer then assembles the edited program, and punches the absolute tape A; compilation and punching of the edited Genie program is done as a separate process.

III.6. AP1 and AP2 coding examples.

The first example which follows is typical of an AP1 program as typed on the Flexowriter; the second example shows the use of AP2 instructions within a Genie program.

```
230                 ORG             Z
                    REM             T4 EQUALS THE SUM OF T5
                                    TO THE POWER I FOR I FROM
                                    ZERO TO J WHERE SL 1, 2,
                                    OR 3 ON INDICATES J
POLY                IF(SLN)SKP      40000
                    TRA             TWO
ONE         T5      FAD             d1.0, U→T4
                    SLF             40000
                    TRA             EXIT
TWO                 IF(SLN)SKP      20000
                    TRA             THREE
            T5      FAD             d1.0
                    FMP             T5
                    FAD             d1.0, U→T4
                    SLF             20000
                    TRA             EXIT
THREE               IF(SLN)SKP      10000
                    TRA             EXIT
            T5      FAD             d1.0
                    FMP             T5
                    FAD             d1.0
                    FMP             T5
                    FAD             d1.0, U→T4
                    SLF             10000
EXIT                TRA             PF
                    END


            DEFINE
            POLY(P,Q).=SEQ
            CC=#ONE if SL^1, #TWO if SL^2, #THREE if SL^3, #END
            BCD     P=SUM Q^I FOR I=0,...,J WHERE SL 1, 2, OR 3
                                    ON INDICATES J
ONE         P=Q+1.0
                    SLF             +40000
            CC=#END
TWO         P=Q^2+Q+1.0
                    SLF             +20000
            CC=#END
THREE       P=Q^3+Q^2+Q+1.0
                    SLF             +10000
END
            DEFINE
LEAVE
```