Chapter 6
What's this Stuff on the Left?

Objectives

Upon completion of this chapter you will be able to:

- Describe the SI instruction format as used with the MVI and CLI instructions,
- Describe the SS instruction format as used with the MVC and CLC instructions,
- Describe the RX instruction format as used with the BC instruction,
- Given a partial .PRN with source code, determine the missing object code, and
- Given a partial .PRN with object code, determine the missing source code.

Introduction

We will now spend some time looking at the assembly listing in detail. For PC/370, this is the .PRN file produced by A370. Our purpose in doing so is to get a better understanding of the assembly process, and to learn about instruction types. Such knowledge is important for debugging purposes. We will even learn to "reverse assemble" a program; that is, given some object code, determine the source instructions.

* * * * * * * * * * * * * * * * * * * *

Each instruction has an instruction type. So far, we have looked at five instructions in detail: MVI, MVC, CLI, CLC, and BC. (Recall that all of the various branch instructions are variations of the BC instruction.) Macros, such as WTO and GET, consist of multiple instructions: they will not be discussed here. Consider the following (admittedly nonsensical) program, STUFF6A.MLC, which uses these instructions:

```
STUFF6A  BEGIN
         CLC   A,B
         BL    D
         MVC   C,B
         MVI   E,C'1'
         B     F
D        EQU   *
         CLI   B,C'D'
         BE    F
         MVC   C,A
         MVI   E,C'9'
F        EQU   *
         RETURN
A        DC    CL3'AB'
B        DC    CL4'CDEF'
C        DS    CL2
E        DC    CL1'*'
         END
```

The following page shows the assembly (.PRN) listing for this program. Those portions of the program which we wish to concentrate on have been shaded (we ignore macro expansions.)

```
 STUFF6A                                               PAGE    1
PC/370 CROSS ASSEMBLER  OPTIONS=LXACE

   LOC               ADR1   ADR2 LINE LABEL    OP       OPERANDS
000000                              1 *++++++  BEGIN
000000                              2 STUFF6A  CSECT
000000                              3          USING    *,15
000000 47F0F058             0058    4          B        KZHQX001
000004 0B                           5          DC       AL1(11)
000005 E2E3E4C6C6F6C140             6          DC       CL11'STUFF6A '
000010 0000000000000000             7 HZQKX001 DC       18F'0'
000058 90ECD00C             000C    8 KZHQX001 STM      14,12,12(13)
00005C 50D0F014             0014    9          ST       13,HZQKX001+4
000060 18ED                        10          LR       14,13
000062 41D0F010             0010   11          LA       13,HZQKX001
000066 50D0E008             0008   12          ST       13,8(0,14)
00006A                             13          DROP     15
00006A                             14          USING    HZQKX001,13
00006A D502D08ED091   009E  00A1   15          CLC      A,B
000070 4740D072             0082   16          BL       D
000074 D201D095D091   00A5  00A1   17          MVC      C,B
00007A 92F1D097             00A7   18          MVI      E,C'1'
00007E 47F0D084             0094   19          B        F
000082               00000082      20 D        EQU      *
000082 95C4D091             00A1   21          CLI      B,C'D'
000086 4780D084             0094   22          BE       F
00008A D201D095D08E   00A5  009E   23          MVC      C,A
000090 92F9D097             00A7   24          MVI      E,C'9'
000094               00000094      25 F        EQU      *
000094                             26 *+++++++ RETURN
000094 58DD0004             0004   27          L        13,4(13)
000098 98ECD00C             000C   28          LM       14,12,12(13)
00009C 07FE                        29          BR       14
00009E C1C240                      30 A        DC       CL3'AB'
0000A1 C3C4C5C6                    31 B        DC       CL4'CDEF'
0000A5                             32 C        DS       CL2
0000A7 5C                          33 E        DC       CL1'*'
0000A8                             34          END
```

The `MVI` and `CLI` instructions are type `SI` instructions, the `MVC` and `CLC` instructions are type `SS` instructions, and the `BC` instruction is a type `RX` instruction. These instruction types are commonly illustrated as follows:

```
       +---------------+-------------+-------------+
       |  1st Halfword | 2nd Halfword | 3rd Halfword |
+------+-------+-------+---+----------+-------------+
|  SI  |Op Code|   I2  |B1 |     D1    |
+------+-------+-------+---+----------+---+----------+
|  SS* |Op Code|   L   |B1 |     D1    |B2 |    D2    |
+------+-------+---+---+---+----------+---+----------+
|  RX  |Op Code|R1 |X2 |B2 |     D2    |
+------+-------+---+---+---+----------+
```

* *There is more than one form of SS instruction. This is the one we will look at for now...*

where:

```
Op Code. .  Operation code
B1, B2 . .  Base register designation field
D1, D2 . .  Displacement field
I2 . . . .  Immediate operand field
L  . . . .  Length field
X2 . . . .  Index Register designation field
```

Notice that *all* instructions use one, two, or three halfwords. Recall that a fullword is four bytes, so a halfword is two bytes. Therefore, *all* instructions occupy two, four, or six bytes.

The SI Instruction Format

```
       +--------------+-------------+-------------+
       |  1st Halfword | 2nd Halfword | 3rd Halfword |
+------+-------+------+---+----------+-------------+
|  SI  |Op Code|  I2  |B1 |    D1    |
+------+-------+------+---+----------+
```

All instructions which use the `SI`, or Storage Immediate, instruction format occupy two halfwords, or four bytes, as shown in the above illustration.

The first of these four bytes is the operation code, or Op Code. Each assembler instruction is assigned a decimal value from 0 to 255, just as are letters and numbers. An instruction may even have the same decimal value as a letter or number! For example, `CLI` is Op Code `X'95'`, which is the same as a lower case `'n'`, and `MVI` is Op Code `X'92'`, which is the same as a lower case `'k'`. The CPU can tell the difference by the context in which it is encountered. Look at our sample program, a portion of which is repeated here. You can see the Op Codes of `95` and `92`.

```
00006A D502D08ED091   009E   00A1    15           CLC      A,B
000070 4740D072              0082    16           BL       D
000074 D201D095D091   00A5   00A1    17           MVC      C,B
00007A 92F1D097              00A7    18           MVI      E,C'1'
00007E 47F0D084              0094    19           B        F
000082             00000082          20 D         EQU      *
000082 95C4D091              00A1    21           CLI      B,C'D'
000086 4780D084              0094    22           BE       F
00008A D201D095D08E   00A5   009E    23           MVC      C,A
000090 92F9D097              00A7    24           MVI      E,C'9'
000094             00000094          25 F         EQU      *
```

The second byte is the immediate value. In the case of the `MVI` and `CLI` instructions, the value to be moved or compared is actually a part of the instruction! For example, in line 18, we want to move a `'1'` to field `E`. The `EBCDIC` equaivalent to a `'1'` is decimal `241` or hex `F1`, and you can see that `F1` is part of the instruction; immediately following the Op Code of `92`. Similarly, in line 21, we want to check to see if the first byte of field `B` is the letter `D`. The `EBCDIC` equivalent to a `'D'` is decimal `196`, or hex `C4`, and you can see that `C4` is part of the instruction; immediately following the Op Code of `95`.

The next part, **base** and **displacement**, is a bit more confusing, but it is something you *must* understand if you are to be successful with assembly language. The next halfword of the instruction indicates the address of the first operand; the field to be moved to or compared to. Within those two bytes, the first half-byte (that's four bits only, or one hex digit) indicates the base register. Recall from our discussion of binary numbers that four bits could range in value from 0 (`0000`, or all bits off) to 15 (`1111`, or all bits on). This is also the range of registers: we have sixteen registers, numbered from 0 to 15 inclusive. So these four bits designate a register. In all three of the above instructions (lines 18, 21, and 24), the next half-byte (following the immediate value) is `D`, which is the hex equivalent to decimal 13. This says that register 13 is pointing to the operands, or close to them. How can this be? Where did we so initialize register 13? We didn't, but the `BEGIN` macro uses register 13 as the **base register**. This is done by the `USING` instruction in line 14, which is part of the expansion of the `BEGIN` macro. Register 13 points to `HZQKX001` (a nonsensical label generated by the `BEGIN` macro) which is the start of the program. The next one and one-half bytes (that's twelve bits) indicate the displacement, or distance, of the operand from `HZQKX001`.

In line 18, the displacement is `X'097'`. The address of `HZQKX001` is `X'000010'` (in the `LOC` column at line 7 of the program). So if we add `X'097'` to `X'000010'` we get `X'0000A7'`, which is the `LOC` for field `E`!

In line 21, the displacement is `X'091'`. The address of `HZQKX001` is `X'000010'` (in the `LOC` column at line 7 of the program). So if we add `X'091'` to `X'000010'` we get `X'0000A1'`, which is the `LOC` for field `B`!

The assembler will save us some of this work. Where approriate, the assembler will place the address of operands in columns `ADR1` or `ADR2`. For example, in line 18, where we wish to move `'1'` to field `E`, `ADR2` is `00A7`, same as we found by adding the displacement to the base. Likewise, in line 21, where we wish to check to see if the first byte of field `B` is the letter `D`, `ADR2` is `00A1`, same as we found above.

Clearly the `ADR1` and `ADR2` columns are easier to use. So why learn the other way? Because when we are looking at object code only, such as during a test session or when debugging, the `ADR1` and `ADR2` columns will not be available.

You Try It...

1. Show the object code for line 18 if that line were changed to `MVI E,C'2'`
2. Show the object code for line 21 if that line were changed to `CLI B+1,C'$'`
3. Show the object code for line 24 if that line were changed to `MVI E+1,X'40'`

The SS Instruction Format

```
              +---------------+-------------+-------------+
              |  1st Halfword | 2nd Halfword | 3rd Halfword |
       +------+------+-------+---+----------+---+----------+
       |  SS  |Op Code|   L   |B1 |    D1    |B2 |    D2    |
       +------+------+-------+---+----------+---+----------+
```

All instructions which use the `SS`, or Storage-to-Storage, instruction format occupy three halfwords, or six bytes, as shown in the above illustration.

As with all assembler instructions, the first of these six bytes is the operation code, or Op Code. `CLC` is Op Code `X'D5'` (same as an upper case `'N'`) and `MVC` is Op Code `X'D2'` (same as an upper case `'K'`). Look at our sample program, a portion of which is repeated here. You can see the Op Codes of `D5` and `D2`.

```
00006A D502D08ED091   009E   00A1   15          CLC       A,B
000070 4740D072              0082   16          BL        D
000074 D201D095D091   00A5   00A1   17          MVC       C,B
00007A 92F1D097              00A7   18          MVI       E,C'1'
00007E 47F0D084              0094   19          B         F
000082              00000082        20 D        EQU       *
000082 95C4D091              00A1   21          CLI       B,C'D'
000086 4780D084              0094   22          BE        F
00008A D201D095D08E   00A5   009E   23          MVC       C,A
000090 92F9D097              00A7   24          MVI       E,C'9'
000094              00000094        25 F        EQU       *
```

The second byte is the length of the operation. Recall that a single byte can range in value from 0 to 255. But a move of length 0 doesn't make any sense. So the length operand is actually the length of the operation *minus one*. In this way, values from 0 to 255 indicate operation lengths (moves, compares, etc.) of from 1 to 256 inclusive. This is why the `MVC` and `CLC` instructions are limited to 256 characters.

In line 15, we compare field `A` with field `B`. Field `A` is defined as 3 bytes long, whereas field `B` is defined as 4 bytes long. Recall from our discussion of the `CLC` instruction that the length of the compare is determined by the length of the first operand. Since that operand, field `A`, is 3 bytes long, 3 bytes will be compared. This is shown in the instruction as 3 *minus one*, or 2. The hex representation of 2 is `X'02'`, and that is what follows the op code; that is, a `CLC` for a length of 3 is represented as `X'D502'`.

In line 17, we move field `B` to field `C`. Field `B` is defined as 4 bytes long, whereas field `C` is defined as 2 bytes long. We know that the length of an `MVC` is determined by the length of the first operand. Since that operand, field `C`, is 2 bytes long, 2 bytes will be compared. This is shown in the instruction as 2 *minus one*, or 1. The hex representation of 1 is `X'01'`, and that is what follows the op code; that is, an `MVC` for a length of 2 is represented as `X'D201'`.

The second halfword (third and fourth bytes) indicates the base and displacement of the first operand, and the third halfword (fifth and sixth bytes) indicates the base and displacement of the second operand. These work the same as described in the `SI` format instruction above, except that there are two operands instead of one. Our earlier discussion of the `ADR1` and `ADR2` columns applies as well.

You Try It...

4. Show the object code for line 15 if that line were changed to `CLC B,A`
5. Show the object code for line 15 if that line were changed to `CLC A+1(2),B+2`
6. Show the object code for line 23 if that line were changed to `MVC C(3),B`

The RX Instruction Format

```
               +---------------+-------------+-------------+
               |  1st Halfword | 2nd Halfword | 3rd Halfword |
        +------+-------+---+---+---+----------+-------------+
        |  RX  |Op Code|R1 |X2 |B2 |    D2    |
        +------+-------+---+---+---+----------+
```

All instructions which use the `RX`, or Register-Index, instruction format occupy two halfwords, or four bytes, as shown in the above illustration.

Again, the first of these four bytes is the operation code. `BC` is Op Code `X'47'`. Look at our sample program: you can see the Op Code of `47`.

```
00006A D502D08ED091   009E  00A1   15          CLC      A,B
000070 4740D072             0082   16          BL       D
000074 D201D095D091   00A5  00A1   17          MVC      C,B
00007A 92F1D097             00A7   18          MVI      E,C'1'
00007E 47F0D084             0094   19          B        F
000082              00000082       20 D        EQU      *
000082 95C4D091             00A1   21          CLI      B,C'D'
000086 4780D084             0094   22          BE       F
00008A D201D095D08E   00A5  009E   23          MVC      C,A
000090 92F9D097             00A7   24          MVI      E,C'9'
000094              00000094       25 F        EQU      *
```

The next half-byte (four bits) corresponds to the mask. The value of this mask, which ranges from `0000` (all bits off) to `1111` (all bits on), is determined by the mnemonic (`BE`, `BL`, etc.) and corresponds to the possible values of the condition code which we discussed in an earlier chapter. For example, we said that if we were comparing two fields, `A` and `B`, and `A` was less than `B`, the resulting condition code would be 0100_2, or `X'4'`. In line 16 above, we want to branch to `D` if `A` is less than `B`. So the mask is `X'4'`, which you see immediately after the op code of `X'47'`.

The next half-byte indicates the index register. An index register is like a displacement whose value depends upon a register. Whereas the other displacements we looked at were constant, the value of the register, and hence the total displacement, may vary. When the index register is 0, it is ignored. This will usually be the case. In all of the above branches, the index register is zero.

We have seen that a `BL` is shown as `X'4740'`, a `B` (unconditional) as `X'47F0'`, and a `BE` as `X'4780'`. For completeness, we show the following table:

BE	X'4780'
BH	X'4720'
BL	X'4740'
BNE	X'4770'
BNH	X'47D0'
BNL	X'47B0'
B	X'47F0'

As before, the second half word of the instruction gives the base and displacement of the label to which the instruction should branch. The only difference between this and the `SI` and `SS` instruction formats which we have discussed is that, again, there is an index register which allows for a second level of displacement.

You Try It...

7. Show the object code for line 16 if that line were changed to `BH D`
8. Show the object code for line 19 if that line were changed to `B D`
9. Show the object code for line 22 if that line were changed to `BNE F`

Data Definition Instructions

Finally, we turn our attention to the data definition instructions, `DS` and `DC`.

```
00009E C1C240                    30 A        DC        CL3'AB'
0000A1 C3C4C5C6                  31 B        DC        CL4'CDEF'
0000A5                           32 C        DS        CL2
0000A7 5C                        33 E        DC        CL1'*'
```

As you can see, when a `DS` instruction is used, no object code is generated: the left side of the listing is blank. However, the location counter is incremented. For example, field `C` begins at location `0000A5`, for a length of two. Therefore, the next field (`E`) will begin at location `0000A5 + 0002 = 0000A7`.

When a `DC` instruction is used, the hexadecimal representation of the data is shown. If the field is over eight bytes in length, the first eight bytes (only) are shown. As with the `DS` instruction, the location counter is incremented. For example, field `A` begin at location `00009E`, for a length of three. Therefore, the next field will begin at location `00009E + 0003 = 0000A1`. Field `A` is initialized to `C'ABb̶'`, which is shown as `X'C1C240'`. *`X'40'` is a blank: memorize that!*

<u>You Try It...</u>

10. Show the object code for line 30 if that line were changed to `A DC CL3'A'`
11. Show the object code for line 31 if that line were changed to `B DC CL4'12'`
12. Show the object code for line 33 if that line were changed to `E DC 2X'FE'`

Exercises

1. True or false.

   T F a. The MVI and MVC instructions are of type SS.
   T F b. Every instruction occupies exactly one, two, or three halfwords.
   T F c. The second byte of an MVI instruction contains the length of the move, minus one.
   T F d. Register 13 is our base register, as indicated by the BEGIN macro.
   T F e. If the first operand of an MVC is indicated by X'D010', then the target (receiving) field begins ten bytes past where register 13 is pointing.
   T F f. If the length of a CLC is indicated by X'0C', then twelve bytes are being compared.
   T F g. The Op Code for BE and BNE are both X'47'.
   T F h. Given E DC C'B', the stuff on the left will be X'C2'.
   T F i. Given F DC CL2'B', the stuff on the left will be X'C2C2'.
   T F j. X'4708' is the object code for BC with a mask of 8.
   T F k. Given A DS CL3 at LOC=0000AE, then A+2 is at LOC=0000B0.
   T F l. Given R DS CL1, the stuff on the left will be X'D9'.
   T F m. The base register designator (B_1 or B_2) occupies one byte of object code.

2. Given MVI D,C'B' generated 92C2D08A at LOC=0000BC, and D is defined as CL3.

   a. What is the LOC for the next instruction?
   b. Show the object code if this line were changed to MVI D,C'C'
   c. Show the object code if this line were changed to MVI D+1,C'D'
   d. Show the object code if this line were changed to MVI D+2,X'DD'
   e. Show the object code if this line were changed to CLI D,C'E'
   f. Show the object code if this line were changed to CLI D+1,B'11100010'

3. Given MVC X,Y generated D20AD090D080 at LOC=0000D6, and Y is defined as CL14.

   a. How long is X?
   b. What is the LOC for the next instruction?
   c. Show the object code if this line were changed to MVC X(4),Y
   d. Show the object code if this line were changed to MVC X+2(5),Y
   e. Show the object code if this line were changed to MVC X+3(1),Y+4
   f. Show the object code if this line were changed to MVC Y,X
   g. Show the object code if this line were changed to CLC X,Y
   h. Show the object code if this line were changed to CLC Y(3),X+1

Exercises

4. a. Fill in the blanks (lines 15-21 and 27):

```
 STUFF6B                                               PAGE    1
PC/370 CROSS ASSEMBLER  OPTIONS=LXACE
   LOC               ADR1    ADR2 LINE LABEL    OP       OPERANDS
000000                               1 *+++++++ BEGIN
000000                               2 STUFF6B  CSECT
000000                               3          USING    *,15
000000 47F0F058              0058    4          B        KZHQX001
000004 0B                            5          DC       AL1(11)
000005 E2E3E4C6C6F6C240              6          DC       CL11'STUFF6B '
000010 0000000000000000              7 HZQKX001 DC       18F'0'
000058 90ECD00C              000C    8 KZHQX001 STM      14,12,12(13)
00005C 50D0F014              0014    9          ST       13,HZQKX001+4
000060 18ED                         10          LR       14,13
000062 41D0F010              0010   11          LA       13,HZQKX001
000066 50D0E008              0008   12          ST       13,8(0,14)
00006A                              13          DROP     15
00006A                              14          USING    HZQKX001,13
00006A D201D086D088   0096   0098   15          MVC      ____,Y
000070 92F9D08D              009D   16          MVI      Z,____
000074 95D3D086              0096   17          CLI      ____,C'L'
000078 4780D07C              008C   18          ____     W
00007C D501D086D08D   0096   009D   19          ____     X,Z
000082 4740D07C              008C   20          BL       ____
000086 D203D08DD088   009D   0098   21          ____     ____,Y
00008C               0000008C       22 W        EQU      *
00008C                              23 *+++++++ RETURN
00008C 58DD0004              0004   24          L        13,4(13)
000090 98ECD00C              000C   25          LM       14,12,12(13)
000094 07FE                         26          BR       14
000096 D1D2                         27 X        DC       CL2'____'
000098 D3D4D5D6D7                   28 Y        DC       CL5'LMNOP'
00009D D8D9E2E3                     29 Z        DC       CL4'QRST'
0000A8                              30          END
```

b. You be the computer...what are the values of `X`, `Y`, and `Z` after this program runs?

X= ________   Y=________   Z=________

Exercises

5. a. Fill in the blanks (lines 15-19 and 25-27):

```
 STUFF6C                                                  PAGE    1
PC/370 CROSS ASSEMBLER  OPTIONS=LXACE
   LOC                ADR1    ADR2 LINE LABEL    OP       OPERANDS
000000                               1 *+++++++ BEGIN
000000                               2 STUFF6C  CSECT
000000                               3          USING    *,15
000000 47F0F058              0058    4          B        KZHQX001
000004 0B                            5          DC       AL1(11)
000005 E2E3E4C6C6F6C340              6          DC       CL11'STUFF6C '
000010 0000000000000000              7 HZQKX001 DC       18F'0'
000058 90ECD00C              000C    8 KZHQX001 STM      14,12,12(13)
00005C 50D0F014              0014    9          ST       13,HZQKX001+4
000060 18ED                         10          LR       14,13
000062 41D0F010              0010   11          LA       13,HZQKX001
000066 50D0E008              0008   12          ST       13,8(0,14)
00006A                              13          DROP     15
00006A                              14          USING    HZQKX001,13
00006A ____D07CD07A   008C   008A   15          CLC      D,F
000070 ____D06C              007C   16          BL       H
000074 ____D07D              008D   17          MVI      E,C'D'
000078 ____D070              0080   18          B        G
00007C ____D07C              008C   19 H        MVI      D,C'F'
000080              00000080        20 G        EQU      *
000080                              21 *+++++++ RETURN
000080 58DD0004              0004   22          L        13,4(13)
000084 98ECD00C              000C   23          LM       14,12,12(13)
000088 07FE                         24          BR       14
00008A C4C5                         25 F        DC       __________
00008C C4                           26 D        DC       __________
00008D C6C5C4                       27 E        DC       __________
000090                              28          END
```

b. You be the computer...what are the values of `F`, `D`, and `E` after this program runs?

`F= ________      D=________      E=________`

Exercises

6. a. Fill in the blanks (lines 15-19):

```
 STUFF6D                                             PAGE    1
PC/370 CROSS ASSEMBLER  OPTIONS=LXACE
   LOC                ADR1    ADR2 LINE LABEL    OP       OPERANDS
000000                               1 *++++++  BEGIN
000000                               2 STUFF6D  CSECT
000000                               3          USING    *,15
000000 47F0F058             0058     4          B        KZHQX001
000004 0B                            5          DC       AL1(11)
000005 E2E3E4C6C6F6C440              6          DC       CL11'STUFF6D '
000010 0000000000000000              7 HZQKX001 DC       18F'0'
000058 90ECD00C             000C     8 KZHQX001 STM      14,12,12(13)
00005C 50D0F014             0014     9          ST       13,HZQKX001+4
000060 18ED                         10          LR       14,13
000062 41D0F010             0010    11          LA       13,HZQKX001
000066 50D0E008             0008    12          ST       13,8(0,14)
00006A                              13          DROP     15
00006A                              14          USING    HZQKX001,13
______ ________             008D    15          MVI      M,C'L'
______ ____________  008A   008D    16          MVC      L(2),M
______ ________             008A    17          CLI      L,C'N'
______ ________             0080    18          BNE      N
______ ________             008A    19          MVI      L,C'M'
000080              00000080        20 N        EQU      *
000080                              21 *+++++++ RETURN
000080 58DD0004             0004    22          L        13,4(13)
000084 98ECD00C             000C    23          LM       14,12,12(13)
000088 07FE                         24          BR       14
00008A D4D6D4                       25 L        DC       CL3'MOM'
00008D 4040                         26 M        DC       CL2'  '
000090                              27          END
```

b. You be the computer...what are the values of `L` and `M` after this program runs?

L= ________   M=________

Exercises

7. a. Fill in the blanks (lines 15-19 and 25-27):

```
 STUFF6E                                              PAGE    1
PC/370 CROSS ASSEMBLER  OPTIONS=LXACE
   LOC                ADR1    ADR2 LINE LABEL    OP       OPERANDS
000000                               1 *++++++  BEGIN
000000                               2 STUFF6E  CSECT
000000                               3          USING    *,15
000000 47F0F058              0058    4          B        KZHQX001
000004 0B                            5          DC       AL1(11)
000005 E2E3E4C6C6F6C540              6          DC       CL11'STUFF6E '
000010 0000000000000000              7 HZQKX001 DC       18F'0'
000058 90ECD00C              000C    8 KZHQX001 STM      14,12,12(13)
00005C 50D0F014              0014    9          ST       13,HZQKX001+4
000060 18ED                         10          LR       14,13
000062 41D0F010              0010   11          LA       13,HZQKX001
000066 50D0E008              0008   12          ST       13,8(0,14)
00006A                              13          DROP     15
00006A                              14          USING    HZQKX001,13
00006A D501D07FD081   008F   0091   15 P        _____    __________
000070 4720D072              0082   16          _____    __________
000074 D201D081D07C   0091   008C   17          _____    __________
00007A 92F2D081              0091   18          _____    __________
00007E 47F0D05A              006A   19          _____    __________
000082              00000082        20 Q        EQU      *
000082                              21 *+++++++ RETURN
000082 58DD0004              0004   22          L        13,4(13)
000086 98ECD00C              000C   23          LM       14,12,12(13)
00008A 07FE                         24          BR       14
00008C F1F2F3                       25 T        _____    __________
00008F F3F3                         26 S        _____    __________
000091 F3F3                         27 R        _____    __________
000098                              28          END
```

b. You be the computer...what are the values of `T`, `S`, and `R` after this program runs?

`T= ________     S=________     R=________`