VisibleZ Lesson 2 – Learning to Access Memory

Now that you can place an address in a register with BASR, you can begin to use the register to point at memory locations and manipulate them. The **Move Characters instruction (MVC)** is used to copy a collection of bytes from one location in memory to another. The instruction has two operands, both of which represent locations in memory. **MVC** has a Storage to Storage (SS1) instruction format which looks like this:

Opcode | L_1L_1 | B_1D_1 |D_1D_1 |B_2D_2 |D_2D_2

The opcode for **MVC** is **d2**. The second byte contains a length that is associated with operand 1. For example, a length of 1C represents a decimal 28 = (1 x 16) + (12). While the length in object code is 28, the actual length is 29 (the machine adds 1). Since the lengths can be expressed as hex digit values from 00 to FF, the actual lengths range from 1 to 256.

The next four digits, $B_1D_1D_1D_1$, represent a base/displacement address for operand 1. The base register is B_1 and $D_1D_1D_1$ represents the displacement. Being 3 hex digits, the displacements can range from x'000' to x'fff' which is 0 to 4095 in decimal. The machine interprets base/displacement addresses by adding the contents of the base register to the displacement. The sum represents the "effective address" in memory. Here is an example:

> Assume register 5 contains x'0000000000001000' and a base/displacement address of 5008
>
> The effective address is computed as x'0000000000001000' + x'008' = x'0000000000001008'

By choosing a base/displacement format for addresses, the object code is smaller than it would be if we coded with absolute addresses. Base/displacement addresses also make it easier to move programs around in memory.

Using VisibleZ, load program **mvc.obj** and cycle past the first BASR, stopping on the first MVC. Here's the object code for MVC: **d2 03 c0 14 c0 18** . The first byte, **d2**, is the opcode. The second byte, **03**, represents the length of operand 1 that represents an actual length of first. As a result, four bytes will be copied from the second operand into the first. The next two bytes, c0 14, represent the base/displacement address of operand 1. How is the effective address computed? The base register is **C** = 12, and we just used BASR to load an address into register 12 (the address is 2). The displacement, 014 is added to 2 producing an effective address of 16.

Operand 1 is the target of the move and the first byte at that address is colored red in VisibleZ. What is the effective address of operand 2? The contents of register 12 is used again and added to the **018** displacement producing an effective address of **1a** . All additions are performed in hexadecimal. When the MVC is cycled, one byte at a time is copied from operand 2 into operand 1 – this is important – the copy operation proceeds one byte at a time. This causes the first operand to fill with b's.

Your second program in VisibleZ

Write a program that contains three fields of length 6. You can place the fields anywhere after the executable code. Fill field 1 with 1's (x'f1'), fill field 2 with 2's, and fill field 3 with spaces (x'40'). The program should swap the contents of field 1 and field 2 using field 3 by coding a sequence of MVCs. Let the last instruction be 07 FF (Branch on Condition Register) which will branch you back to the beginning instruction.

Learning more

System/z machines are designed to use a varying number of bits of a 64 bit register (2^{64} is a mind-bogglingly large number) when computing effective addresses. In this tri-modal architecture the number of digits can be 24, 31, or 64. Older machines computed with 24-bit addresses. This provided an address space of 2^{24} = 16 megabytes. VisibleZ (like many programs today) uses the rightmost 31-bits of a register which provides an address space with

2^{31} = 2 gigabytes (also called gibibytes). A 64-bit address provides for an address space of 2^{64} = 16 exbibytes. Someone estimated that all words ever spoken by human beings is approximately 5 exabytes (exabytes are smaller than exbibytes).