#SHAREorg

VisibleZ: a Freeware Product for Teaching IBM Assembler and System/z Architecture

Dr. David E. Woolbright
Columbus State University

August 7, 2012
Session Number: 11793

Who Am I?

- Dr. David E. Woolbright
  - Professor of Computer Science
  - Columbus State University
  - Columbus, Georgia
  - Email: woolbright_david@columbusstate.edu
  - Assembler Blog: www.punctiliousprogrammer.com

Who Are You?

- Someone who needs to learn IBM assembly language?
- Someone who needs to teach IBM assembly language?
- A Java programmer interesting in exploring IBM assembly language?

My goals today

- To introduce you to the features of VisibleZ
- To suggest three ways VisizbleZ can help you learn (or teach) assembly language
- To try out a few of the many VisibleZ lessons that can ease you into assembly language

What is VisibleZ?

- Freeware
- An object code interpreter
- A visualization tool for watching instructions execute on a System/z machine
- A tool for learning new assembler instructions
- A tool for teaching IBM instruction architecture
- A collection of Java classes

Where can you get the software?

- http://csc.columbusstate.edu/woolbright/visiblez.xml

Related Websites

- http://www.punctiliousprogrammer.com

  - An assembler website
- http://csc.columbusstate.edu/woolbright

  - My academic website

What’s on the websites?

- Product download
- General articles about programming assembly language (Base/Displacement Addressing, DSECTs, Looping,…)
- Articles about specific instructions (semantics and programming tips)
- A video course (in development)
- VisibleZ lessons
- An assembler blog

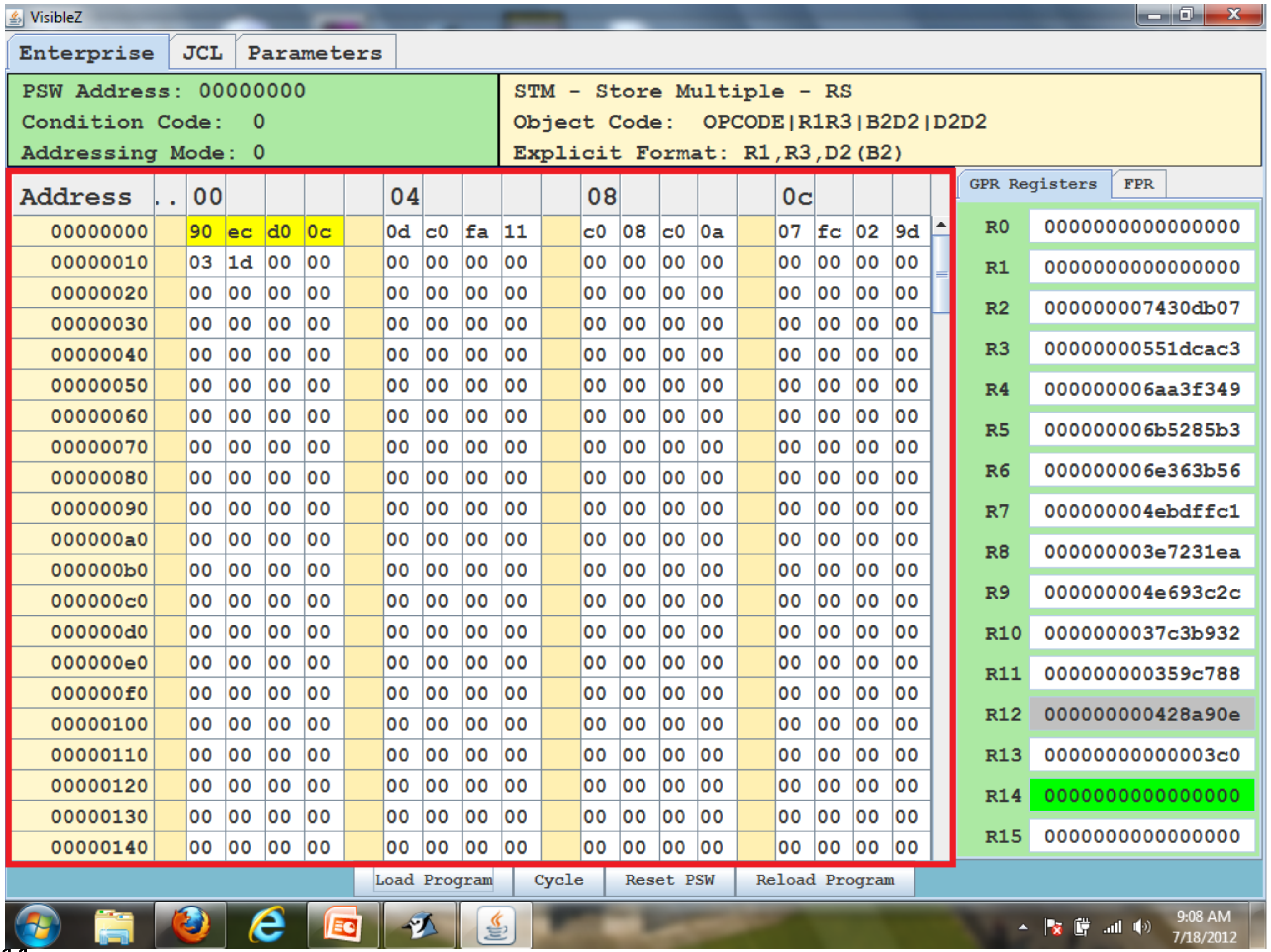
What’s included in the product download?

- A BlueJ project with lots of Java code (> 100 classes)
- A \Codes directory with one or more object code programs for every supported instruction

Versions

- Desktop – 32 bytes per row in the memory display
- Low Res – 16 bytes per row
- Android Pad (Honeycomb) – in development
- A version with complete source code
- A version with partial source code
- An executable jar version

What’s on the main panel?
…. Memory dump



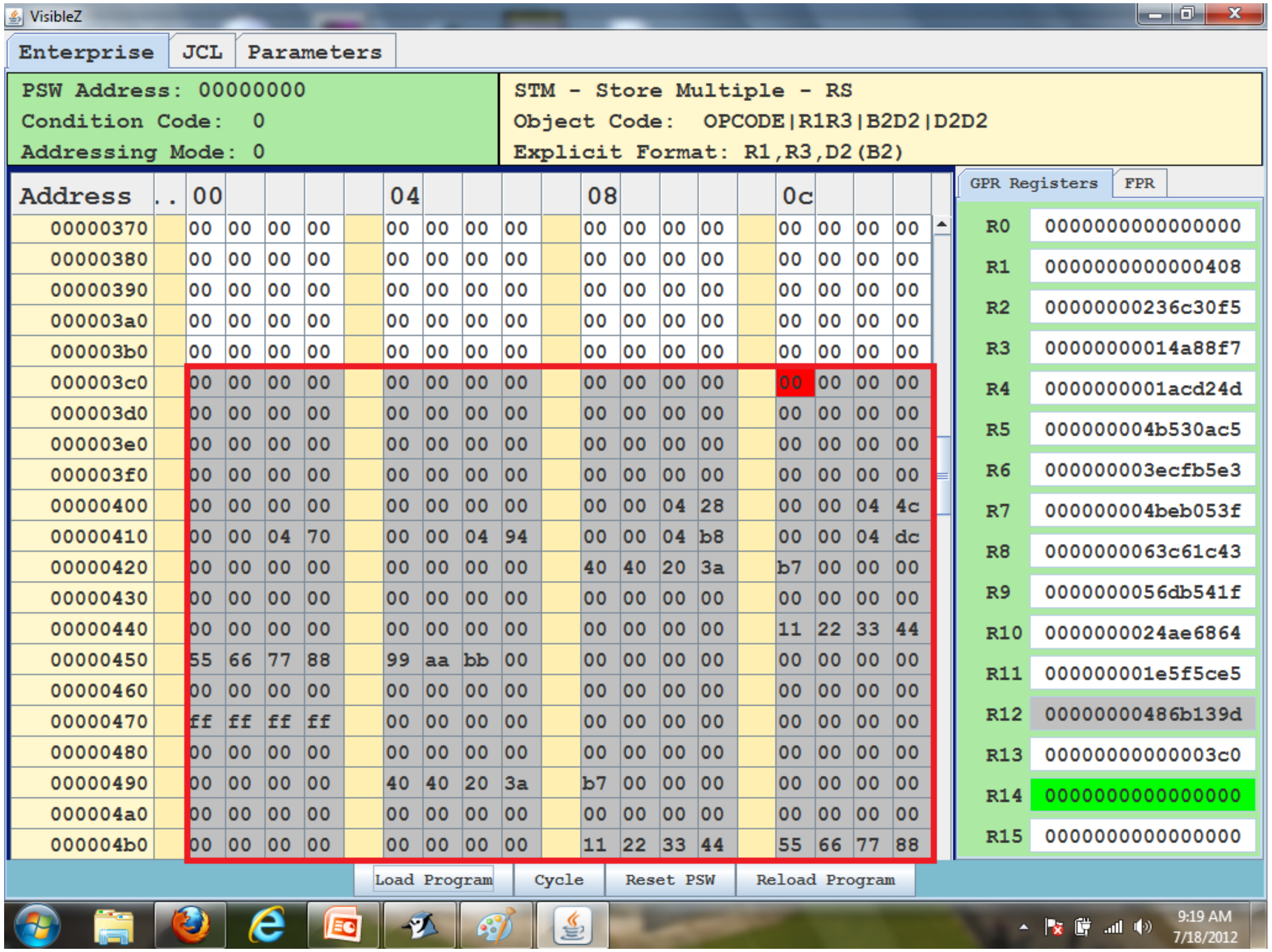
```
PSW Address:  00000000
Condition Code:   0
Addressing Mode:  0
```

```
STM - Store Multiple - RS
Object Code:  OPCODE|R1R3|B2D2|D2D2
Explicit Format: R1,R3,D2(B2)
```

```
Address  ..  00           04           08           0c
00000000     90 ec d0 0c  0d c0 fa 11  c0 08 c0 0a  07 fc 02 9d
00000010     03 1d 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000020     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000030     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000040     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000050     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000060     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000070     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000080     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000090     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
000000a0     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
000000b0     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
000000c0     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
000000d0     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
000000e0     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
000000f0     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000100     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000110     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000120     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000130     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000140     00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
```

```
R0   0000000000000000
R1   0000000000000000
R2   000000007430db07
R3   00000000551dcac3
R4   000000006aa3f349
R5   000000006b5285b3
R6   000000006e363b56
R7   000000004ebdffc1
R8   000000003e7231ea
R9   000000004e693c2c
R10  0000000037c3b932
R11  000000000359c788
R12  000000000428a90e
R13  00000000000003c0
R14  0000000000000000
R15  0000000000000000
```

A small operating system area



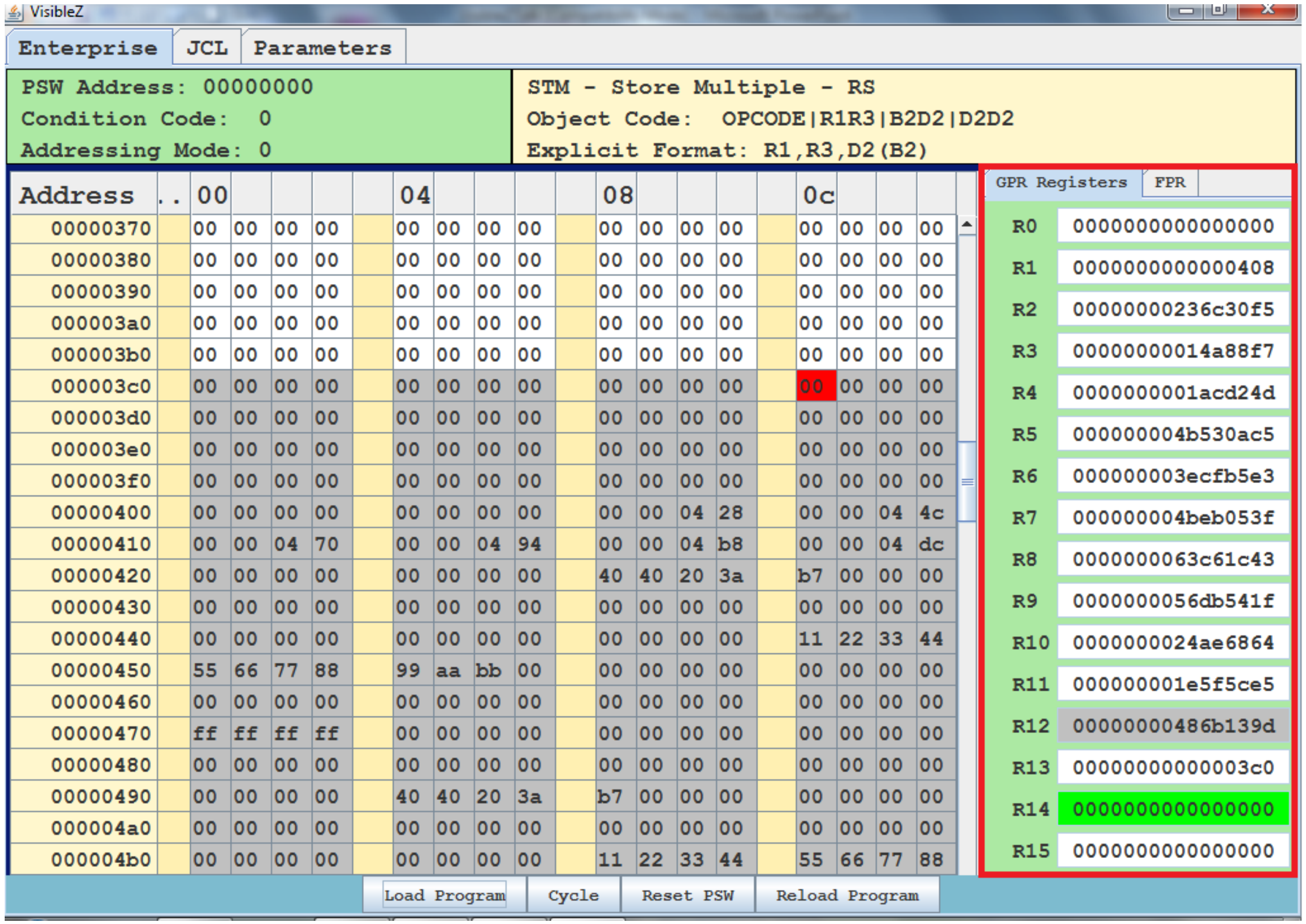
```
PSW Address:  00000000
Condition Code:   0
Addressing Mode:  0
```

```
STM - Store Multiple - RS
Object Code:  OPCODE|R1R3|B2D2|D2D2
Explicit Format: R1,R3,D2(B2)
```

```
Address   ..  00          04          08          0c
00000370      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000380      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000390      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003a0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003b0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003c0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003d0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003e0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003f0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000400      00 00 00 00 00 00 00 00 00 00 04 28 00 00 04 4c
00000410      00 00 04 70 00 00 04 94 00 00 04 b8 00 00 04 dc
00000420      00 00 00 00 00 00 00 00 40 40 20 3a b7 00 00 00
00000430      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000440      00 00 00 00 00 00 00 00 00 00 00 00 11 22 33 44
00000450      55 66 77 88 99 aa bb 00 00 00 00 00 00 00 00 00
00000460      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000470      ff ff ff ff 00 00 00 00 00 00 00 00 00 00 00 00
00000480      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000490      00 00 00 00 40 40 20 3a b7 00 00 00 00 00 00 00
000004a0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000004b0      00 00 00 00 00 00 00 00 11 22 33 44 55 66 77 88
```

```
GPR Registers   FPR
R0   0000000000000000
R1   0000000000000408
R2   00000000236c30f5
R3   0000000014a88f7
R4   000000001acd24d
R5   000000004b530ac5
R6   000000003ecfb5e3
R7   000000004beb053f
R8   0000000063c61c43
R9   0000000056db541f
R10  0000000024ae6864
R11  000000001e5f5ce5
R12  0000000486b139d
R13  00000000000003c0
R14  0000000000000000
R15  0000000000000000
```

Registers

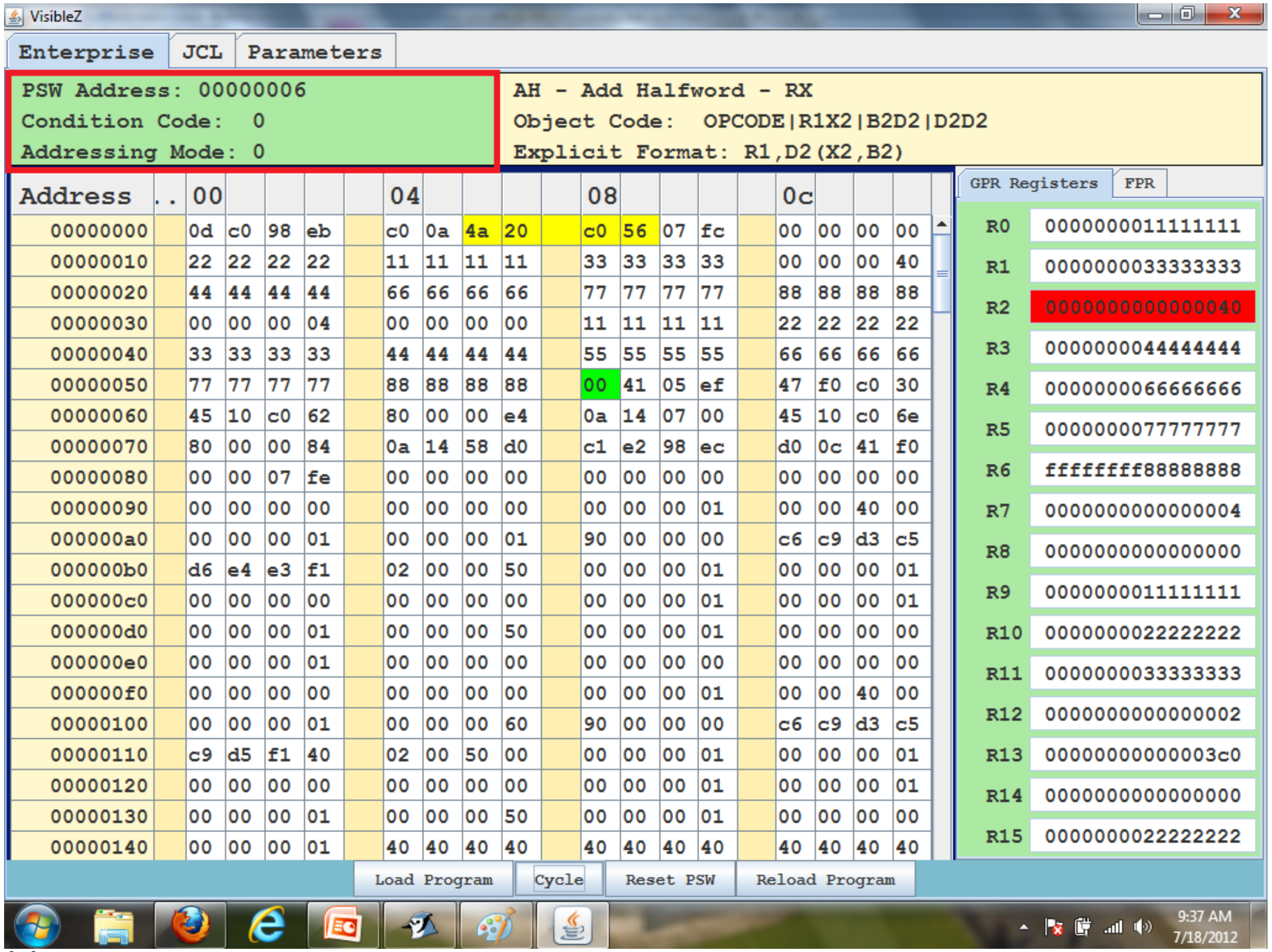


```
PSW Address:  00000000          STM - Store Multiple - RS
Condition Code:  0              Object Code:  OPCODE|R1R3|B2D2|D2D2
Addressing Mode:  0             Explicit Format:  R1,R3,D2(B2)
```

```
Address   ..  00          04          08          0c
00000370      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000380      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000390      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003a0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003b0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003c0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003d0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003e0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003f0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000400      00 00 00 00 00 00 00 00 00 00 04 28 00 00 04 4c
00000410      00 00 04 70 00 00 04 94 00 00 04 b8 00 00 04 dc
00000420      00 00 00 00 00 00 00 00 40 40 20 3a b7 00 00 00
00000430      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000440      00 00 00 00 00 00 00 00 00 00 00 00 11 22 33 44
00000450      55 66 77 88 99 aa bb 00 00 00 00 00 00 00 00 00
00000460      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000470      ff ff ff ff 00 00 00 00 00 00 00 00 00 00 00 00
00000480      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000490      00 00 00 00 40 40 20 3a b7 00 00 00 00 00 00 00
000004a0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000004b0      00 00 00 00 00 00 00 00 11 22 33 44 55 66 77 88
```

```
R0   0000000000000000
R1   0000000000000408
R2   00000000236c30f5
R3   0000000014a88f7
R4   000000001acd24d
R5   000000004b530ac5
R6   000000003ecfb5e3
R7   000000004beb053f
R8   0000000063c61c43
R9   0000000056db541f
R10  000000024ae6864
R11  000000001e5f5ce5
R12  0000000486b139d
R13  00000000000003c0
R14  0000000000000000
R15  0000000000000000
```

Program Status Word



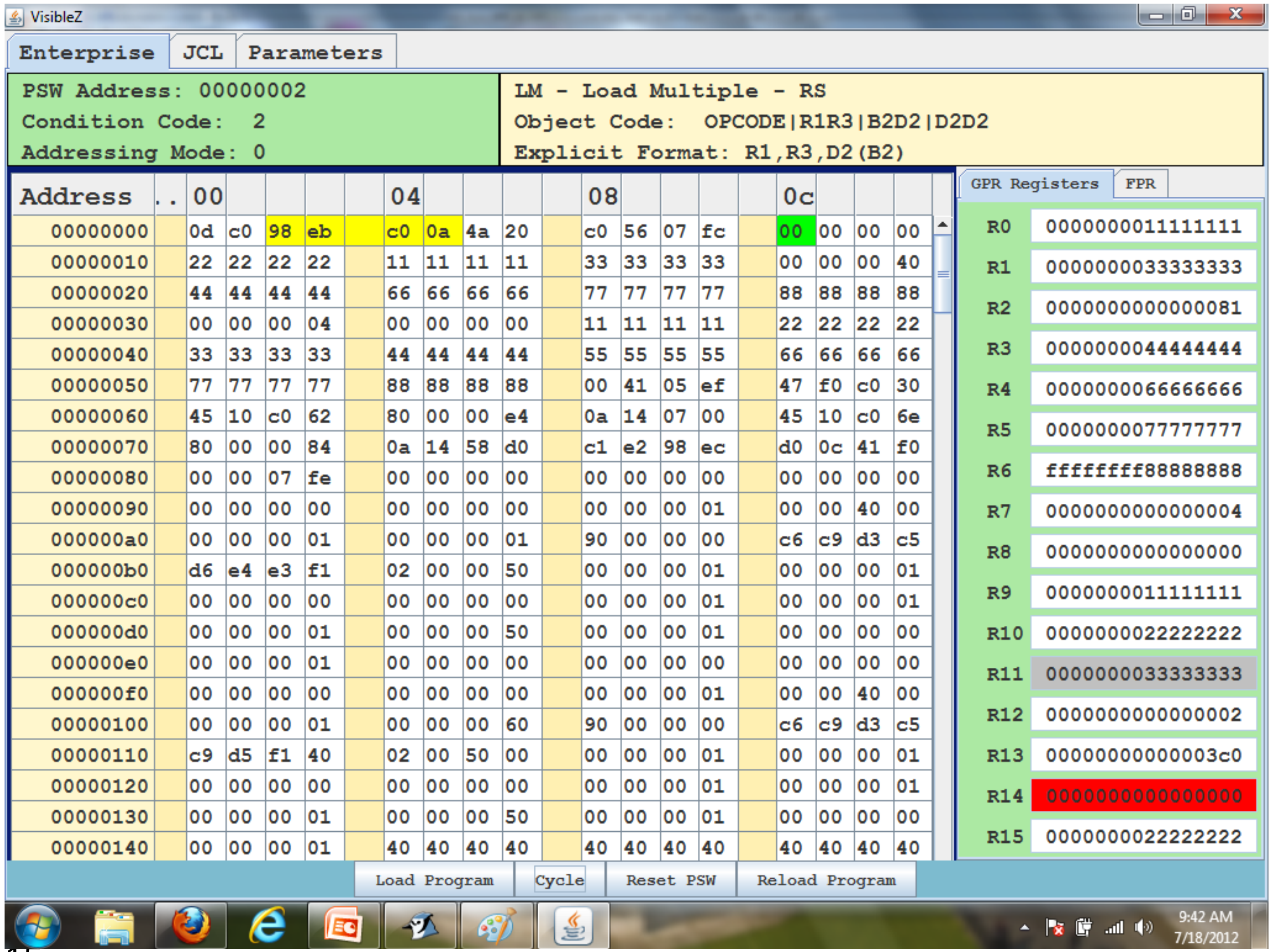
PSW Address: 00000006
Condition Code: 0
Addressing Mode: 0

AH - Add Halfword - RX
Object Code: OPCODE|R1X2|B2D2|D2D2
Explicit Format: R1,D2(X2,B2)

Address	..	00				04				08				0c			
00000000		0d	c0	98	eb	c0	0a	4a	20	c0	56	07	fc	00	00	00	00
00000010		22	22	22	22	11	11	11	11	33	33	33	33	00	00	00	40
00000020		44	44	44	44	66	66	66	66	77	77	77	77	88	88	88	88
00000030		00	00	00	04	00	00	00	00	11	11	11	11	22	22	22	22
00000040		33	33	33	33	44	44	44	44	55	55	55	55	66	66	66	66
00000050		77	77	77	77	88	88	88	88	00	41	05	ef	47	f0	c0	30
00000060		45	10	c0	62	80	00	00	e4	0a	14	07	00	45	10	c0	6e
00000070		80	00	00	84	0a	14	58	d0	c1	e2	98	ec	d0	0c	41	f0
00000080		00	00	07	fe	00	00	00	00	00	00	00	00	00	00	00	00
00000090		00	00	00	00	00	00	00	00	00	00	00	01	00	00	40	00
000000a0		00	00	00	01	00	00	00	01	90	00	00	00	c6	c9	d3	c5
000000b0		d6	e4	e3	f1	02	00	00	50	00	00	00	01	00	00	00	01
000000c0		00	00	00	00	00	00	00	00	00	00	00	01	00	00	00	01
000000d0		00	00	00	01	00	00	00	50	00	00	00	01	00	00	00	00
000000e0		00	00	00	01	00	00	00	00	00	00	00	00	00	00	00	00
000000f0		00	00	00	00	00	00	00	00	00	00	00	01	00	00	40	00
00000100		00	00	00	01	00	00	00	60	90	00	00	00	c6	c9	d3	c5
00000110		c9	d5	f1	40	02	00	50	00	00	00	00	01	00	00	00	01
00000120		00	00	00	00	00	00	00	00	00	00	00	01	00	00	00	01
00000130		00	00	00	01	00	00	00	50	00	00	00	01	00	00	00	00
00000140		00	00	00	01	40	40	40	40	40	40	40	40	40	40	40	40

R0	0000000011111111
R1	0000000033333333
R2	0000000000000040
R3	0000000044444444
R4	0000000066666666
R5	0000000077777777
R6	ffffffff88888888
R7	0000000000000004
R8	0000000000000000
R9	0000000011111111
R10	0000000022222222
R11	0000000033333333
R12	0000000000000002
R13	00000000000003c0
R14	0000000000000000
R15	0000000022222222

Color coding of instructions



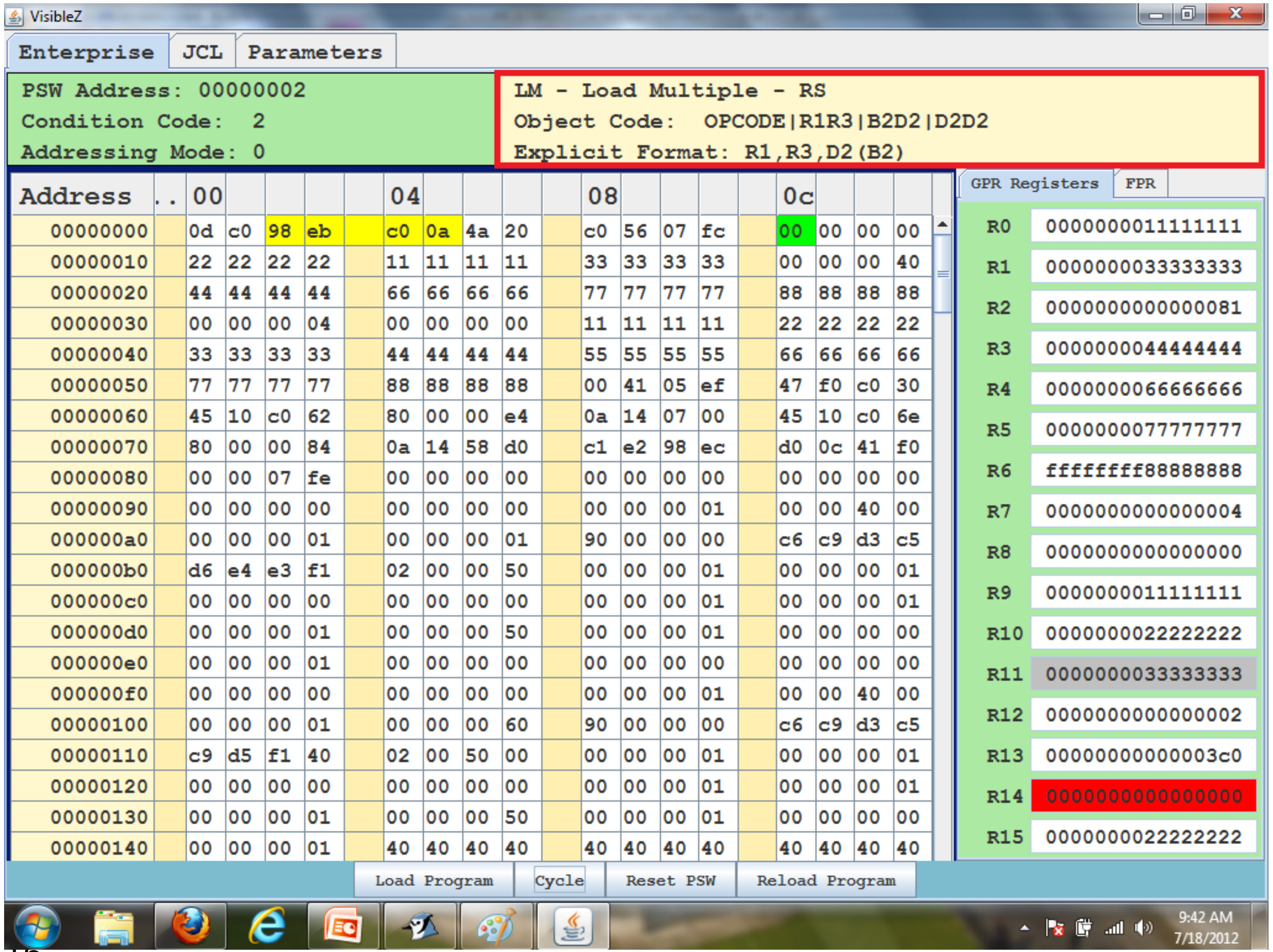
PSW Address: 00000002
Condition Code: 2
Addressing Mode: 0

LM - Load Multiple - RS
Object Code: OPCODE|R1R3|B2D2|D2D2
Explicit Format: R1,R3,D2(B2)

Address	..	00				04				08				0c			
00000000		0d	c0	98	eb	c0	0a	4a	20	c0	56	07	fc	00	00	00	00
00000010		22	22	22	22	11	11	11	11	33	33	33	33	00	00	00	40
00000020		44	44	44	44	66	66	66	66	77	77	77	77	88	88	88	88
00000030		00	00	00	04	00	00	00	00	11	11	11	11	22	22	22	22
00000040		33	33	33	33	44	44	44	44	55	55	55	55	66	66	66	66
00000050		77	77	77	77	88	88	88	88	00	41	05	ef	47	f0	c0	30
00000060		45	10	c0	62	80	00	00	e4	0a	14	07	00	45	10	c0	6e
00000070		80	00	00	84	0a	14	58	d0	c1	e2	98	ec	d0	0c	41	f0
00000080		00	00	07	fe	00	00	00	00	00	00	00	00	00	00	00	00
00000090		00	00	00	00	00	00	00	00	00	00	00	01	00	00	40	00
000000a0		00	00	00	01	00	00	00	01	90	00	00	00	c6	c9	d3	c5
000000b0		d6	e4	e3	f1	02	00	00	50	00	00	00	01	00	00	00	01
000000c0		00	00	00	00	00	00	00	00	00	00	00	01	00	00	00	01
000000d0		00	00	00	01	00	00	00	50	00	00	00	01	00	00	00	00
000000e0		00	00	00	01	00	00	00	00	00	00	00	00	00	00	00	00
000000f0		00	00	00	00	00	00	00	00	00	00	00	01	00	00	40	00
00000100		00	00	00	01	00	00	00	60	90	00	00	00	c6	c9	d3	c5
00000110		c9	d5	f1	40	02	00	50	00	00	00	00	01	00	00	00	01
00000120		00	00	00	00	00	00	00	00	00	00	00	01	00	00	00	01
00000130		00	00	00	01	00	00	00	50	00	00	00	01	00	00	00	00
00000140		00	00	00	01	40	40	40	40	40	40	40	40	40	40	40	40

R0	0000000011111111
R1	0000000033333333
R2	0000000000000081
R3	0000000044444444
R4	0000000066666666
R5	0000000077777777
R6	ffffffff88888888
R7	0000000000000004
R8	0000000000000000
R9	0000000011111111
R10	0000000022222222
R11	0000000033333333
R12	0000000000000002
R13	00000000000003c0
R14	0000000000000000
R15	0000000022222222

Information about the current instruction



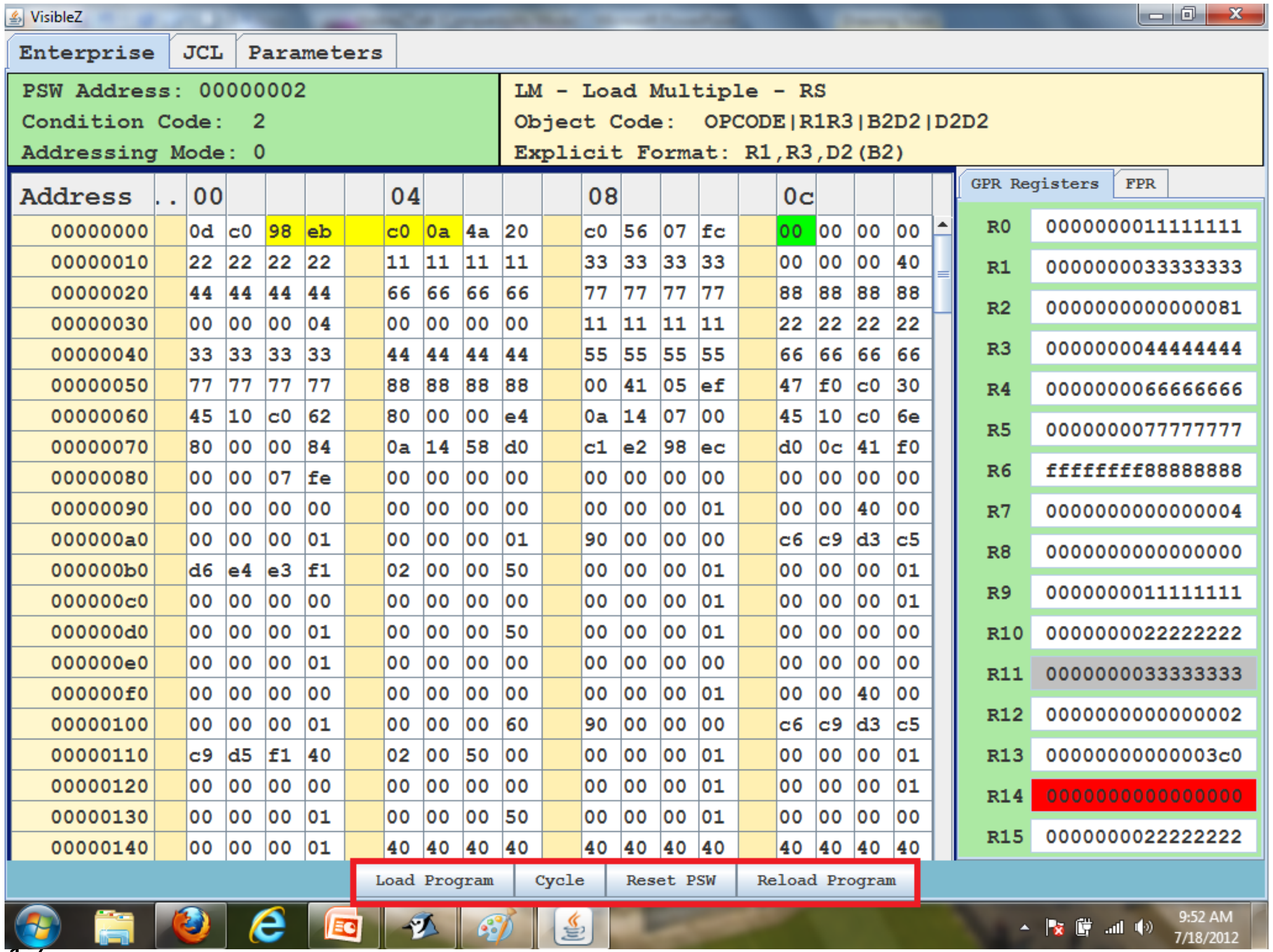
```
PSW Address:  00000002
Condition Code:   2
Addressing Mode:  0
```

```
LM - Load Multiple - RS
Object Code:   OPCODE|R1R3|B2D2|D2D2
Explicit Format:  R1,R3,D2(B2)
```

```
Address   ..  00                   04                   08                   0c
00000000      0d  c0  98  eb       c0  0a  4a  20       c0  56  07  fc       00  00  00  00
00000010      22  22  22  22       11  11  11  11       33  33  33  33       00  00  00  40
00000020      44  44  44  44       66  66  66  66       77  77  77  77       88  88  88  88
00000030      00  00  00  04       00  00  00  00       11  11  11  11       22  22  22  22
00000040      33  33  33  33       44  44  44  44       55  55  55  55       66  66  66  66
00000050      77  77  77  77       88  88  88  88       00  41  05  ef       47  f0  c0  30
00000060      45  10  c0  62       80  00  00  e4       0a  14  07  00       45  10  c0  6e
00000070      80  00  00  84       0a  14  58  d0       c1  e2  98  ec       d0  0c  41  f0
00000080      00  00  07  fe       00  00  00  00       00  00  00  00       00  00  00  00
00000090      00  00  00  00       00  00  00  00       00  00  00  01       00  00  40  00
000000a0      00  00  00  01       00  00  00  01       90  00  00  00       c6  c9  d3  c5
000000b0      d6  e4  e3  f1       02  00  00  50       00  00  00  01       00  00  00  01
000000c0      00  00  00  00       00  00  00  00       00  00  00  01       00  00  00  01
000000d0      00  00  00  01       00  00  00  50       00  00  00  01       00  00  00  00
000000e0      00  00  00  01       00  00  00  00       00  00  00  00       00  00  00  00
000000f0      00  00  00  00       00  00  00  00       00  00  00  01       00  00  40  00
00000100      00  00  00  01       00  00  00  60       90  00  00  00       c6  c9  d3  c5
00000110      c9  d5  f1  40       02  00  50  00       00  00  00  01       00  00  00  01
00000120      00  00  00  00       00  00  00  00       00  00  00  01       00  00  00  01
00000130      00  00  00  01       00  00  00  50       00  00  00  01       00  00  00  00
00000140      00  00  00  01       40  40  40  40       40  40  40  40       40  40  40  40
```

```
R0    0000000011111111
R1    0000000033333333
R2    0000000000000081
R3    0000000044444444
R4    0000000066666666
R5    0000000077777777
R6    ffffffff88888888
R7    0000000000000004
R8    0000000000000000
R9    0000000011111111
R10   0000000022222222
R11   0000000033333333
R12   0000000000000002
R13   00000000000003c0
R14   0000000000000000
R15   0000000022222222
```

Load, Cycle, Reset, Reload Buttons



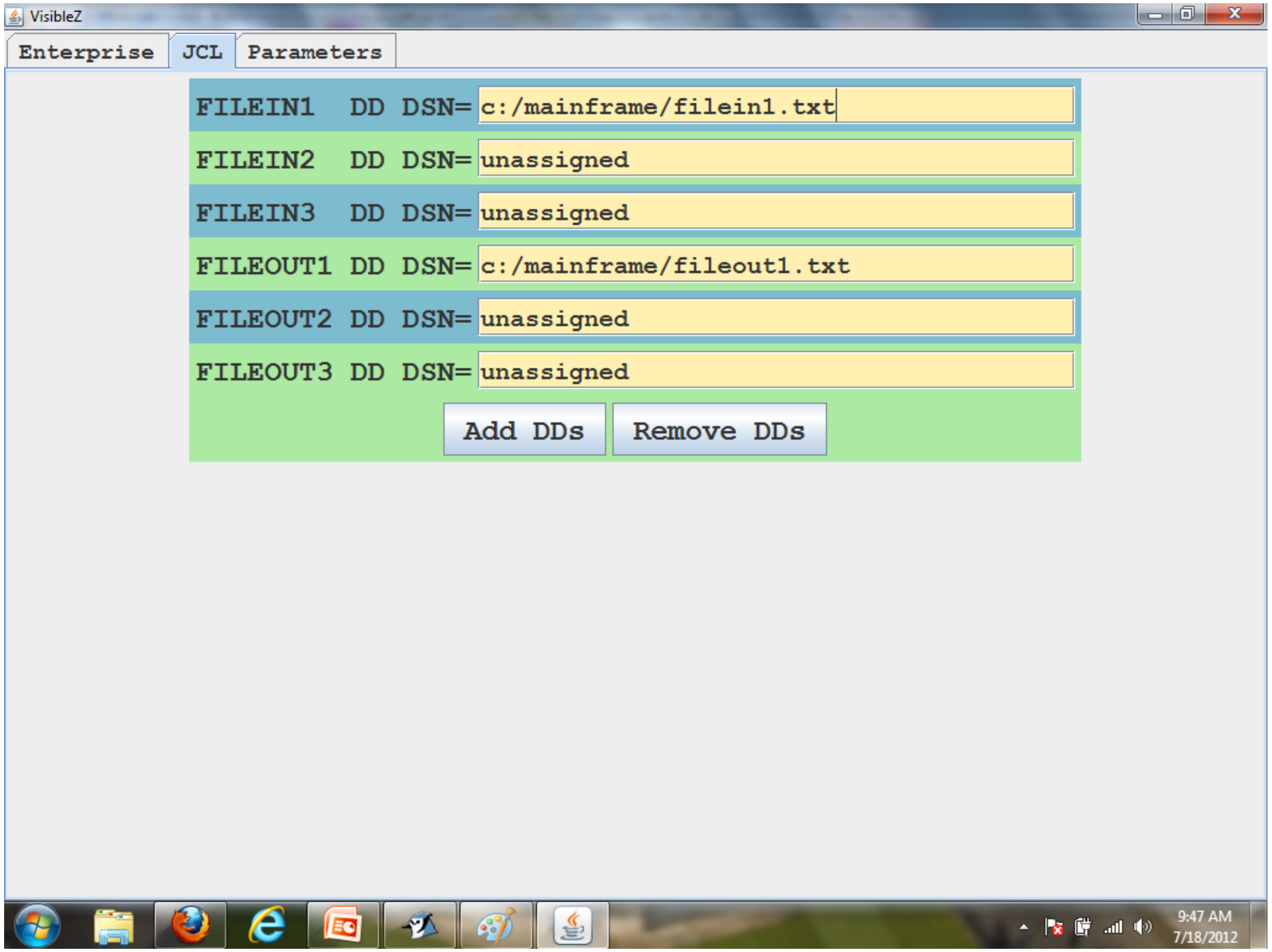
PSW Address: 00000002
Condition Code: 2
Addressing Mode: 0

LM - Load Multiple - RS
Object Code: OPCODE|R1R3|B2D2|D2D2
Explicit Format: R1,R3,D2(B2)

Address	..	00					04					08					0c			
00000000		0d	c0	98	eb		c0	0a	4a	20		c0	56	07	fc		00	00	00	00
00000010		22	22	22	22		11	11	11	11		33	33	33	33		00	00	00	40
00000020		44	44	44	44		66	66	66	66		77	77	77	77		88	88	88	88
00000030		00	00	00	04		00	00	00	00		11	11	11	11		22	22	22	22
00000040		33	33	33	33		44	44	44	44		55	55	55	55		66	66	66	66
00000050		77	77	77	77		88	88	88	88		00	41	05	ef		47	f0	c0	30
00000060		45	10	c0	62		80	00	00	e4		0a	14	07	00		45	10	c0	6e
00000070		80	00	00	84		0a	14	58	d0		c1	e2	98	ec		d0	0c	41	f0
00000080		00	00	07	fe		00	00	00	00		00	00	00	00		00	00	00	00
00000090		00	00	00	00		00	00	00	00		00	00	00	01		00	00	40	00
000000a0		00	00	00	01		00	00	00	01		90	00	00	00		c6	c9	d3	c5
000000b0		d6	e4	e3	f1		02	00	00	50		00	00	00	01		00	00	00	01
000000c0		00	00	00	00		00	00	00	00		00	00	00	01		00	00	00	01
000000d0		00	00	00	01		00	00	00	50		00	00	00	01		00	00	00	00
000000e0		00	00	00	01		00	00	00	00		00	00	00	00		00	00	00	00
000000f0		00	00	00	00		00	00	00	00		00	00	00	01		00	00	40	00
00000100		00	00	00	01		00	00	00	60		90	00	00	00		c6	c9	d3	c5
00000110		c9	d5	f1	40		02	00	50	00		00	00	00	01		00	00	00	01
00000120		00	00	00	00		00	00	00	00		00	00	00	01		00	00	00	01
00000130		00	00	00	01		00	00	00	50		00	00	00	01		00	00	00	00
00000140		00	00	00	01		40	40	40	40		40	40	40	40		40	40	40	40

R0	0000000011111111
R1	0000000033333333
R2	0000000000000081
R3	0000000044444444
R4	0000000066666666
R5	0000000077777777
R6	ffffffff88888888
R7	0000000000000004
R8	0000000000000000
R9	0000000011111111
R10	0000000022222222
R11	0000000033333333
R12	0000000000000002
R13	00000000000003c0
R14	0000000000000000
R15	0000000022222222

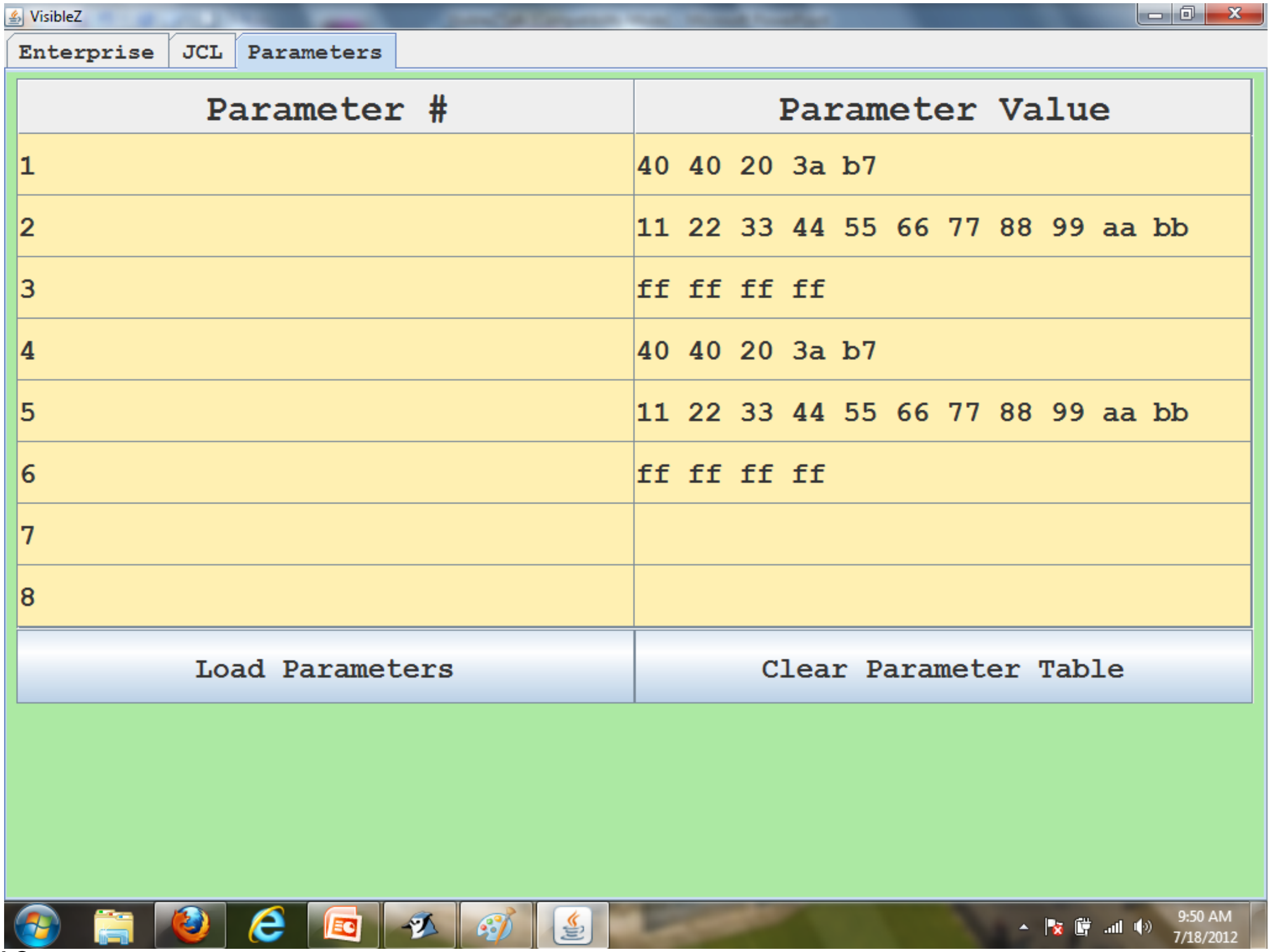
Simple File Support


VisibleZ

Enterprise | JCL | Parameters

```
FILEIN1  DD DSN= c:/mainframe/filein1.txt
FILEIN2  DD DSN= unassigned
FILEIN3  DD DSN= unassigned
FILEOUT1 DD DSN= c:/mainframe/fileout1.txt
FILEOUT2 DD DSN= unassigned
FILEOUT3 DD DSN= unassigned
```

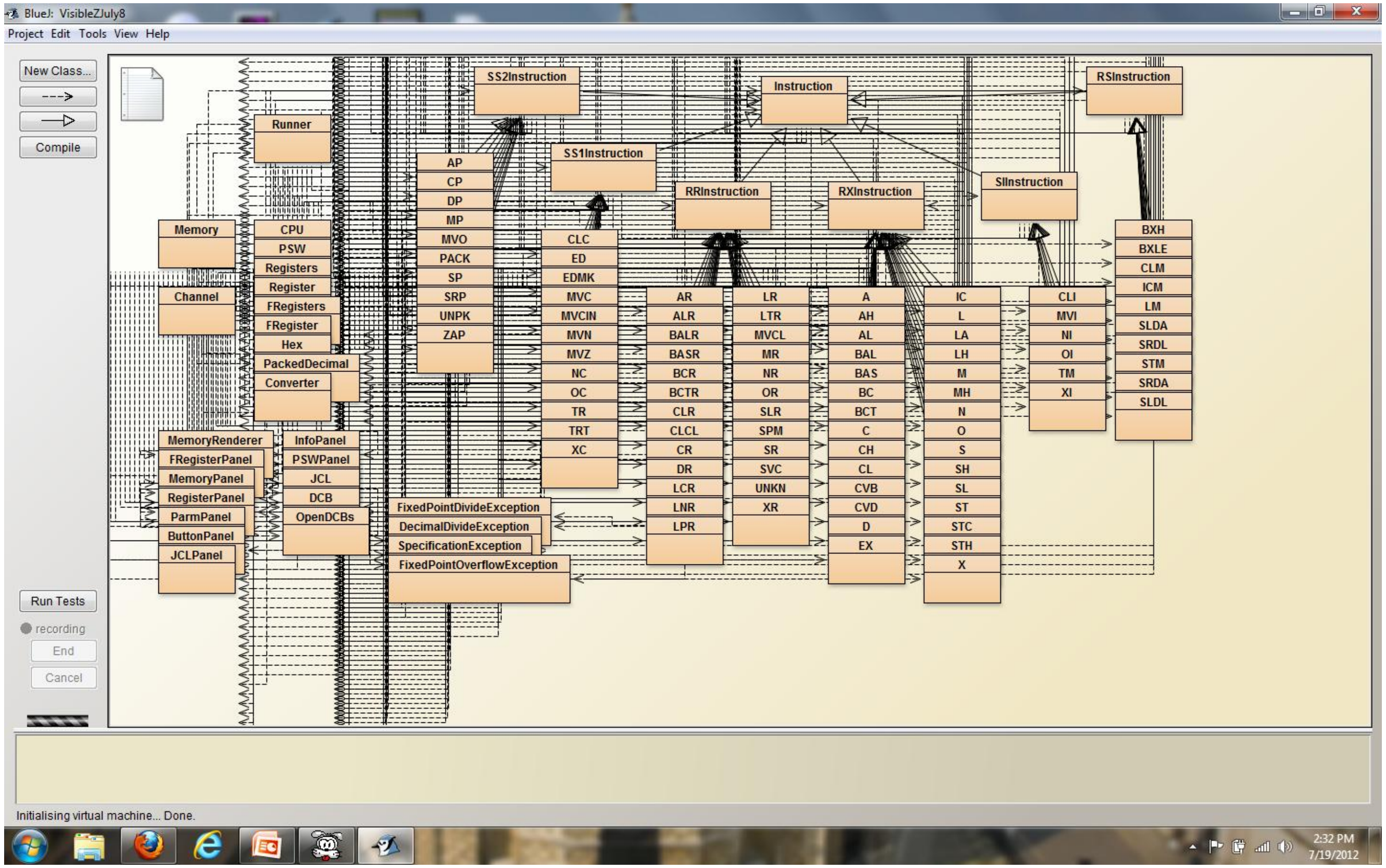
Parameter Passing



VisibleZ

Parameter #	Parameter Value
1	40 40 20 3a b7
2	11 22 33 44 55 66 77 88 99 aa bb
3	ff ff ff ff
4	40 40 20 3a b7
5	11 22 33 44 55 66 77 88 99 aa bb
6	ff ff ff ff
7	
8	

VisibleZ is Object-oriented



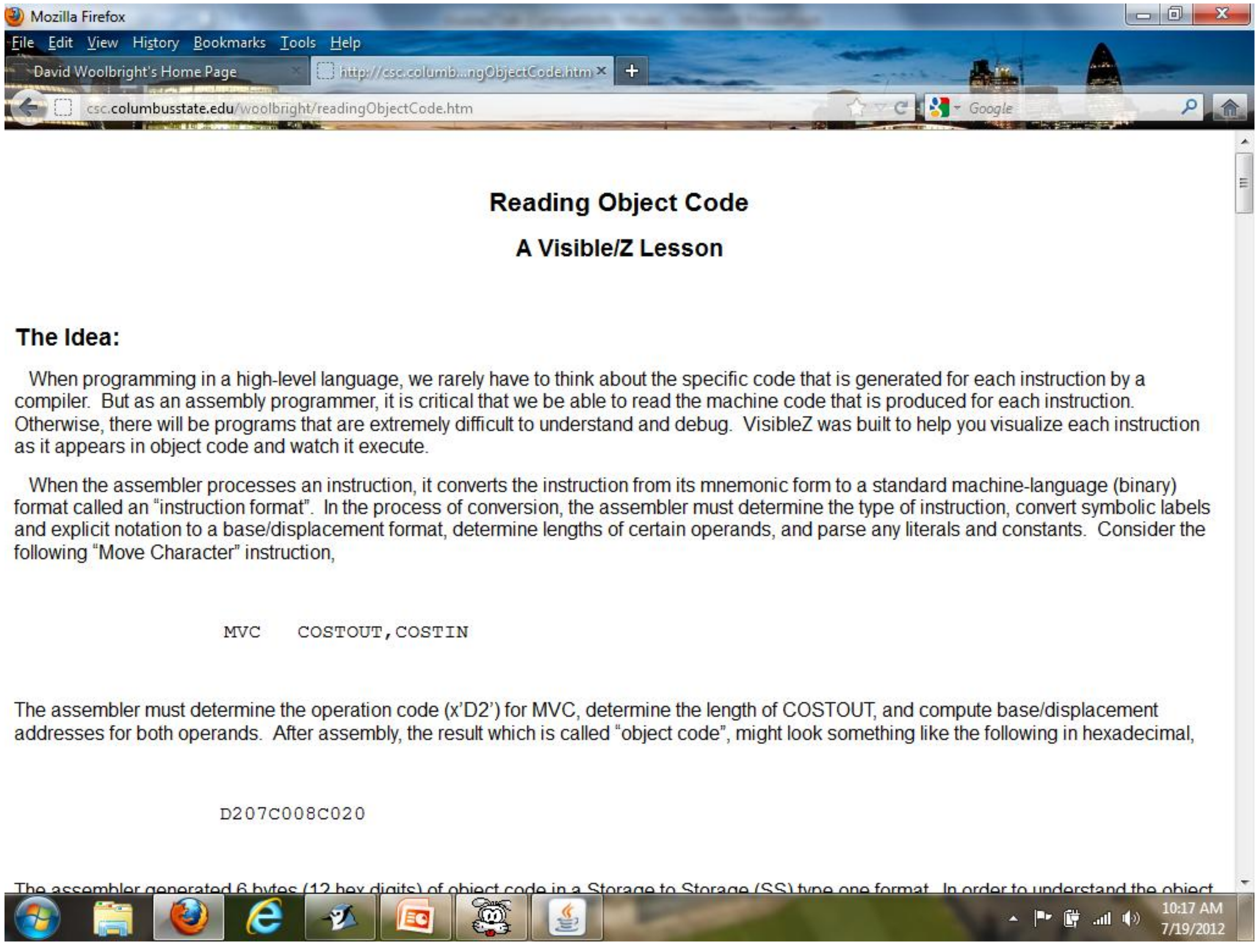
Each instruction is a Class with an execute( ) method

```
public void execute()
{
 // Instructions have access to Register, Memory, PSW, …
//  objects
// The execute( ) method manipulates the objects to
// provide the semantics of each instruction
}
```

How VisibleZ can help: Strategy 1

- The product is distributed with small object code programs that demonstrate how each instruction works
- There are also object code programs that target fundamental assembler concepts
- VisibleZ lessons cover general programming concepts and specific instructions
- Read an article on the website, pick a program, load it up, and watch it work

A VisibleZ Lesson: Reading Object Code


Mozilla Firefox
File Edit View History Bookmarks Tools Help
David Woolbright's Home Page
http://csc.columb...ngObjectCode.htm
csc.columbusstate.edu/woolbright/readingObjectCode.htm
Google
10:17 AM
7/19/2012

Reading Object Code

A Visible/Z Lesson

The Idea:

When programming in a high-level language, we rarely have to think about the specific code that is generated for each instruction by a compiler. But as an assembly programmer, it is critical that we be able to read the machine code that is produced for each instruction. Otherwise, there will be programs that are extremely difficult to understand and debug. VisibleZ was built to help you visualize each instruction as it appears in object code and watch it execute.

When the assembler processes an instruction, it converts the instruction from its mnemonic form to a standard machine-language (binary) format called an "instruction format". In the process of conversion, the assembler must determine the type of instruction, convert symbolic labels and explicit notation to a base/displacement format, determine lengths of certain operands, and parse any literals and constants. Consider the following "Move Character" instruction,

```
MVC    COSTOUT,COSTIN
```

The assembler must determine the operation code (x'D2') for MVC, determine the length of COSTOUT, and compute base/displacement addresses for both operands. After assembly, the result which is called "object code", might look something like the following in hexadecimal,

```
D207C008C020
```

The assembler generated 6 bytes (12 hex digits) of object code in a Storage to Storage (SS) type one format. In order to understand the object

Why are instruction formats important?

- Each instruction format encapsulates all the information that is available to the CPU when an instruction is executed
- Example:

```
MVC    X,Y
Opcode | LL1 | B1D | DD | B2D | DD
  D2   |  03 |  C0 | 04 |  C0 | 0E
```

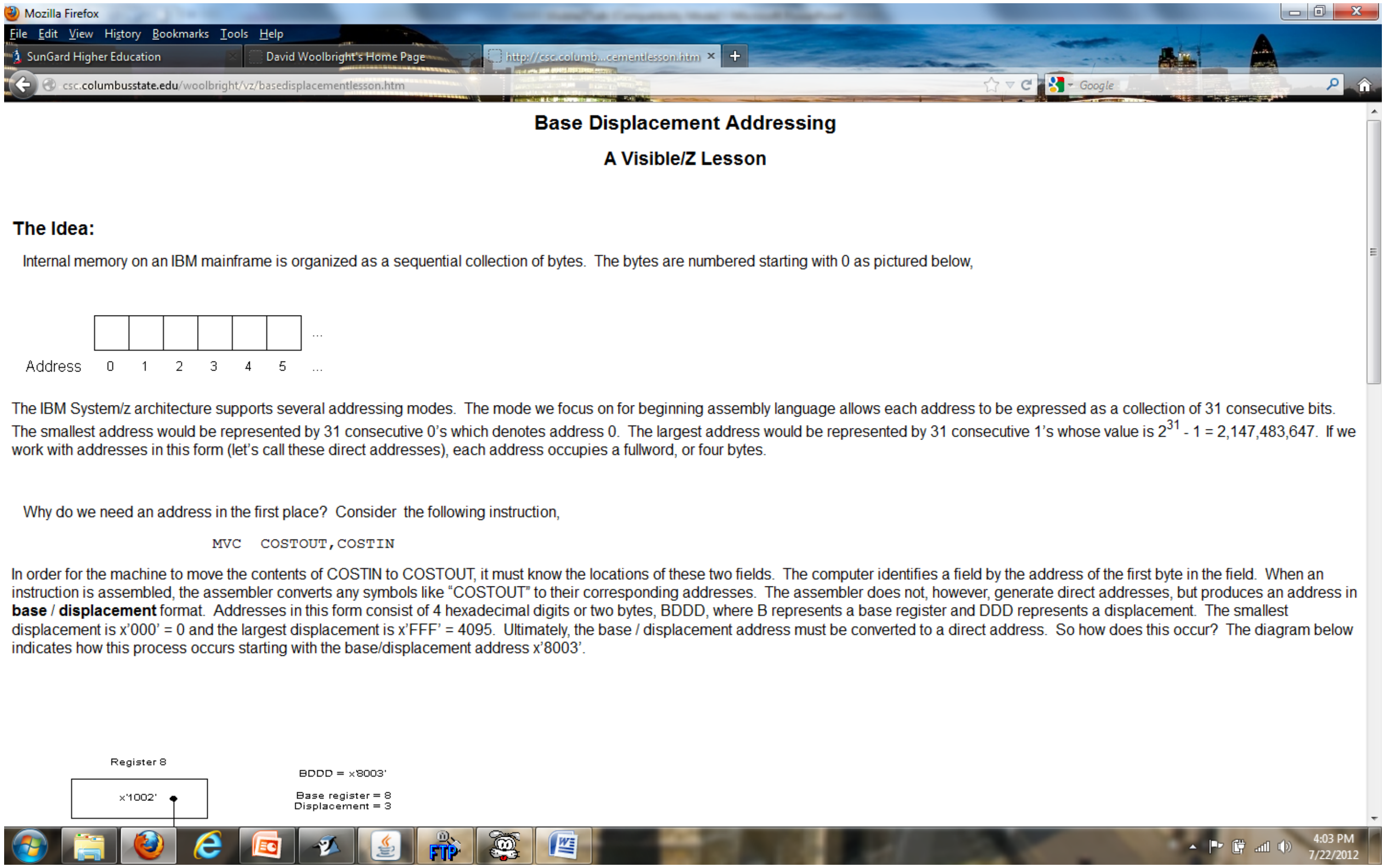
- What does the CPU know?

  Operation, length, Beginning address of X and Y
- What is missing?

Using VisibleZ: Reading Object Code

- Load the program **readingobjectcode.obj**
- Step through each instruction
- Each instruction format is presented with each instruction and will help you master the five/six basic instruction formats

A VisibleZ Lesson: Base Displacement Addressing


Mozilla Firefox
File Edit View History Bookmarks Tools Help
SunGard Higher Education
David Woolbright's Home Page
http://csc.columb...cementlesson.htm
csc.columbusstate.edu/woolbright/vz/basedisplacementlesson.htm
Google

Base Displacement Addressing

A Visible/Z Lesson

The Idea:

Internal memory on an IBM mainframe is organized as a sequential collection of bytes. The bytes are numbered starting with 0 as pictured below,

The IBM System/z architecture supports several addressing modes. The mode we focus on for beginning assembly language allows each address to be expressed as a collection of 31 consecutive bits. The smallest address would be represented by 31 consecutive 0's which denotes address 0. The largest address would be represented by 31 consecutive 1's whose value is $2^{31} - 1 = 2,147,483,647$. If we work with addresses in this form (let's call these direct addresses), each address occupies a fullword, or four bytes.

Why do we need an address in the first place? Consider the following instruction,

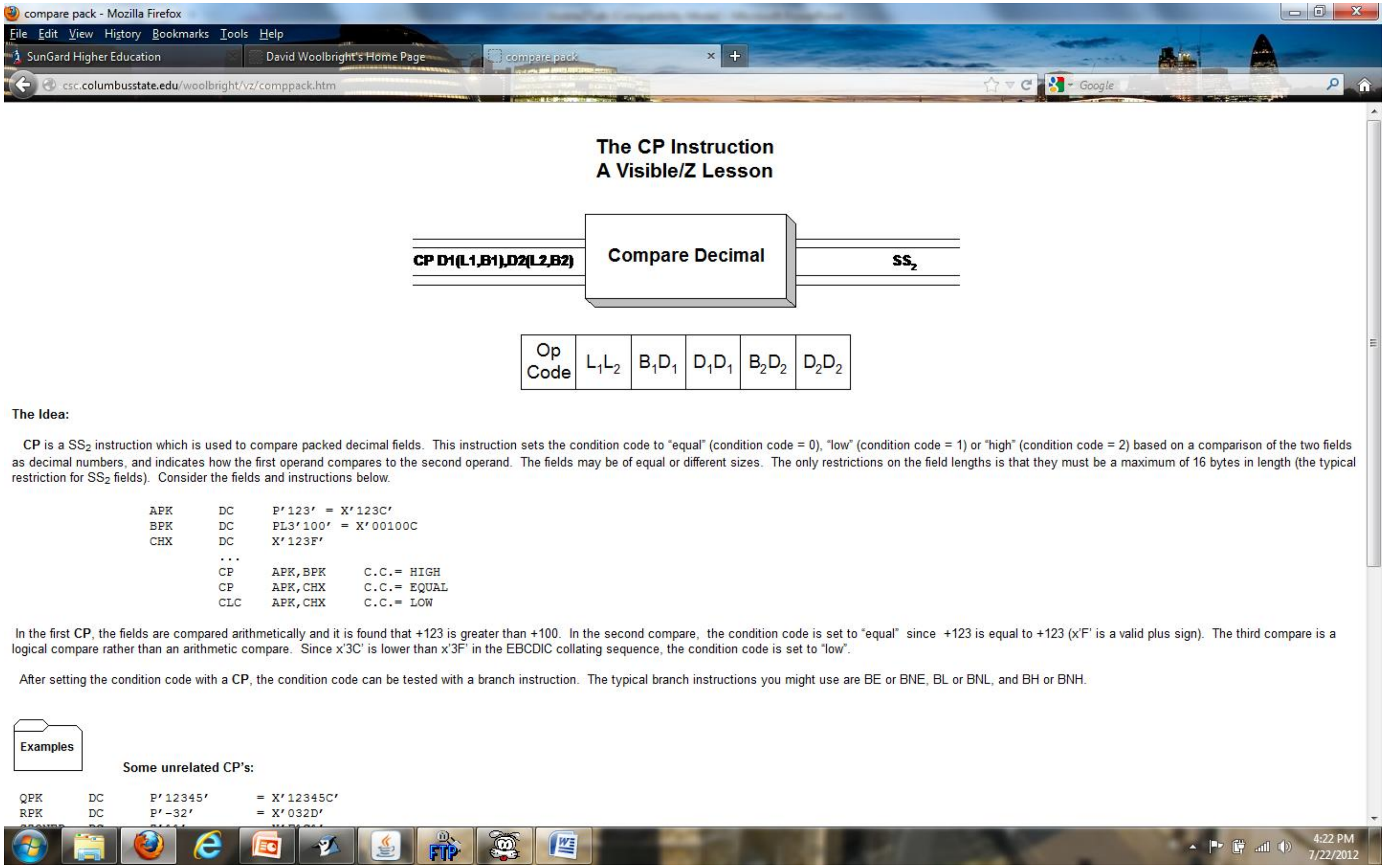
```
MVC   COSTOUT,COSTIN
```

In order for the machine to move the contents of COSTIN to COSTOUT, it must know the locations of these two fields. The computer identifies a field by the address of the first byte in the field. When an instruction is assembled, the assembler converts any symbols like "COSTOUT" to their corresponding addresses. The assembler does not, however, generate direct addresses, but produces an address in **base** / **displacement** format. Addresses in this form consist of 4 hexadecimal digits or two bytes, BDDD, where B represents a base register and DDD represents a displacement. The smallest displacement is x'000' = 0 and the largest displacement is x'FFF' = 4095. Ultimately, the base / displacement address must be converted to a direct address. So how does this occur? The diagram below indicates how this process occurs starting with the base/displacement address x'8003'.

Using VisibleZ: Loading a Base Register

- Load the program **baseregister.obj**
- Step through each instruction
- Load the program **baseregister1.obj**
- How can the same instruction address different fields?

A VisibleZ Lesson: The CP Instruction


compare pack - Mozilla Firefox
File Edit View History Bookmarks Tools Help
SunGard Higher Education
David Woolbright's Home Page
compare pack
csc.columbusstate.edu/woolbright/vz/comppack.htm
Google
4:22 PM
7/22/2012

The CP Instruction
A Visible/Z Lesson

CP D1(L1,B1),D2(L2,B2)
Compare Decimal
SS2

Op Code	L_1L_2	B_1D_1	D_1D_1	B_2D_2	D_2D_2

The Idea:

CP is a SS_2 instruction which is used to compare packed decimal fields. This instruction sets the condition code to "equal" (condition code = 0), "low" (condition code = 1) or "high" (condition code = 2) based on a comparison of the two fields as decimal numbers, and indicates how the first operand compares to the second operand. The fields may be of equal or different sizes. The only restrictions on the field lengths is that they must be a maximum of 16 bytes in length (the typical restriction for SS_2 fields). Consider the fields and instructions below.

```
APK      DC      P'123'  = X'123C'
BPK      DC      PL3'100'  = X'00100C
CHX      DC      X'123F'
         ...
         CP      APK,BPK      C.C.= HIGH
         CP      APK,CHX      C.C.= EQUAL
         CLC     APK,CHX      C.C.= LOW
```

In the first **CP**, the fields are compared arithmetically and it is found that +123 is greater than +100. In the second compare, the condition code is set to "equal" since +123 is equal to +123 (x'F' is a valid plus sign). The third compare is a logical compare rather than an arithmetic compare. Since x'3C' is lower than x'3F' in the EBCDIC collating sequence, the condition code is set to "low".

After setting the condition code with a **CP**, the condition code can be tested with a branch instruction. The typical branch instructions you might use are BE or BNE, BL or BNL, and BH or BNH.

Examples

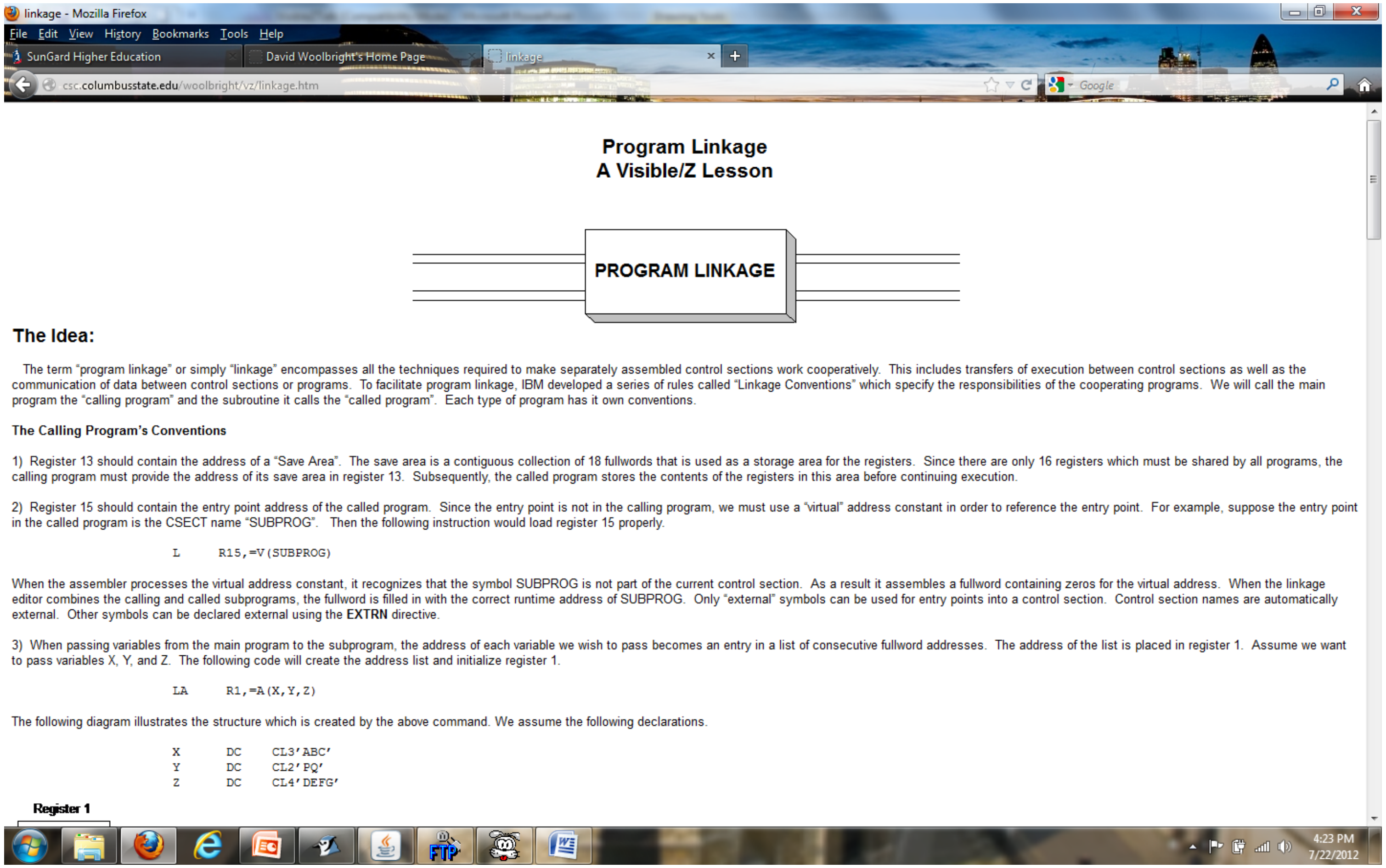
Some unrelated CP's:

```
QPK      DC      P'12345'      = X'12345C'
RPK      DC      P'-32'        = X'032D'
```

Using VisibleZ: The CP Instruction

- Load the program **cp.obj**
- Step through each instruction
- Repeat for programs **cp1.obj**, **cp2.obj** and **cp3.obj**

A VisibleZ Lesson:
Program Linkage and Parameter Passing


linkage - Mozilla Firefox
File Edit View History Bookmarks Tools Help
SunGard Higher Education
David Woolbright's Home Page
linkage
csc.columbusstate.edu/woolbright/vz/linkage.htm
Google
4:23 PM
7/22/2012

Program Linkage
A Visible/Z Lesson

PROGRAM LINKAGE

The Idea:

The term "program linkage" or simply "linkage" encompasses all the techniques required to make separately assembled control sections work cooperatively. This includes transfers of execution between control sections as well as the communication of data between control sections or programs. To facilitate program linkage, IBM developed a series of rules called "Linkage Conventions" which specify the responsibilities of the cooperating programs. We will call the main program the "calling program" and the subroutine it calls the "called program". Each type of program has it own conventions.

The Calling Program's Conventions

1) Register 13 should contain the address of a "Save Area". The save area is a contiguous collection of 18 fullwords that is used as a storage area for the registers. Since there are only 16 registers which must be shared by all programs, the calling program must provide the address of its save area in register 13. Subsequently, the called program stores the contents of the registers in this area before continuing execution.

2) Register 15 should contain the entry point address of the called program. Since the entry point is not in the calling program, we must use a "virtual" address constant in order to reference the entry point. For example, suppose the entry point in the called program is the CSECT name "SUBPROG". Then the following instruction would load register 15 properly.

```
         L       R15,=V(SUBPROG)
```

When the assembler processes the virtual address constant, it recognizes that the symbol SUBPROG is not part of the current control section. As a result it assembles a fullword containing zeros for the virtual address. When the linkage editor combines the calling and called subprograms, the fullword is filled in with the correct runtime address of SUBPROG. Only "external" symbols can be used for entry points into a control section. Control section names are automatically external. Other symbols can be declared external using the **EXTRN** directive.

3) When passing variables from the main program to the subprogram, the address of each variable we wish to pass becomes an entry in a list of consecutive fullword addresses. The address of the list is placed in register 1. Assume we want to pass variables X, Y, and Z. The following code will create the address list and initialize register 1.

```
         LA      R1,=A(X,Y,Z)
```

The following diagram illustrates the structure which is created by the above command. We assume the following declarations.

```
         X       DC      CL3'ABC'
         Y       DC      CL2'PQ'
         Z       DC      CL4'DEFG'
```

Register 1

Using VisibleZ: Passing Parameters

- Load the program **linkage.obj**
- Step through each instruction
- Repeat for programs **linkage1.obj**, and **linkage2.obj**

How VisibleZ can help: Strategy 2

- Write or modify object code programs to exercise the instructions you are studying
- Let’s try this: Write an object code program that moves a field X to a field Y

How VisibleZ can help: Strategy 3

- Pick an instruction.
- Study the instruction semantics in Principles of Operation.
- Code the Java implementation of the instruction in the execute() method of the instruction class
- VisibleZ is distributed in two versions (with and without all the instruction semantics)
- Try this: Delete the code in the execute( ) method (or start with the empty version) of MVC and provide it yourself.

Help me improve VisibleZ

- I'd like to hear from you about how to make the product better
- E-mail: woolbright_david@columbusstate.edu
- Thanks for listening!