# CSI International®
# Modernizing VSE:
# With z/Cobol™.

John Rankin

October 23th-25th, 2023
17th European GSE
Stuttgart, Germany

CSI INTERNATIONAL

# The State of Cobol VSE/ESA

- Hardware Support
  - ✓ ESA instructions, limited in their ability to use current facilities
  - ✓ Heavily designed towards memory to memory operation
- Software Support
  - ✓ Old design for code base
    - Built upon 1968 code structure
    - Limited ability to react to hardware changes
    - Designed for a stable architecture
    - Discarded and rewritten for Enterprise Cobol
  - ✓ Use of runtime and assembler objects
    - Internal routines are used from the 1968 version of Cobol
    - LE used for runtime, limits performance in exchange for support

# z/Cobol™ for VSE

- z/Cobol™ for VSE is designed for all modern IBM Mainframes
  - ✓ 100% Compatible for IBM Cobol VSE/ESA 1.1, 21csw 1.2
  - ✓ Meets the National Institute of Standards and Technology
    - 100% Certified against ANSI 1985 Cobol Standard
    - 340,000 Lines of Cobol, Testing Code Suite with 500+ programs tests
  - ✓ Maximum Performance on all IBM z Series mainframe platforms
- Completely New
- Generates High Performance Code
- Built for z/VSE and VSE$^{n}$

# Compiler Operation

- Cobol Source code can be structured to match:
  - ✓ ANSI 1968, ANSI 1974, ANSI 1985, or ISO/IEC 1989:2014(E)
  - ✓ Enhancements specifically designed for VSE
    - TCP/IP for ANSI Communication Description Entries
    - Report Description Entries, Compliant with ANSI 1974 and ANSI 1985
    - Addition VSE based file methods, and all IBM extensions
- Generated Object code:
  - ✓ Linked with provided library of non LE routines.
  - ✓ No LE environment required.
  - ✓ Object code can be executed anywhere in VSE
- Supports fully functional export of assembler code as output

# Code Generated

- 100% 64 bit.
  - ✓ Dynamically adjusts to the calling environment, and uses IBM z/OS save area structures
- Reentrant, no SIIS issues
- Optimized for non memory to memory operation
- Takes full advantage of Long Displacements and Relative Branching
  - ✓ Each storage section can be addressed 1 Megabyte at a time
  - ✓ Branching works with full word relative movements
- All supporting routines
  - ✓ Provided in linkable 64bit object decks
  - ✓ Designed to work with 24/31/64 objects where necessary
- Runtime error recovery, including vector displays

# Intellectual Property
# z/Cobol™ for VSE

- U.S. Patents:
  - ✓ 10,901,739 Issued: January 26th, 2021
  - ✓ 11,429,390 Issued: August 20th, 2022
- COBOL Standards
  - ✓ CODASYL COBOL – Issued: July, 1968
  - ✓ X3.23-1974 ANSI COBOL, Issued: May 10th, 1974
  - ✓ X3.23-1985 ANSI COBOL, Issued: September 10th, 1985
  - ✓ ISO/IEC 1989/Amendment 1, Intrinsic function module

# Embracing the Power of the Hardware

- Levels of Support
  - ✓ Basic z/Series support, Architecture level 5
    - 64 instructions, relative branching, and long displacements
  - ✓ Immediate Values, Architecture level 7
    - Reduces the use of literal pools, and moves literals into instructions
  - ✓ Decimal Floating Point, Architecture level 9
    - Moves binary coded decimal away from memory to memory operations
  - ✓ Vector Facility, Architecture 12
    - Uses all 32 Vector Registers
    - Completely eliminates binary coded decimal memory instructions
    - All math operations occur on the chip, storing only when necessary

# Vector Code Generated

- As math operations occur when z/Cobol™ elements are loaded into Vectors
- All 32 Vectors are continually used
- Vectors are saved prior to call operations
  - ✓ Allows z/Cobol™ code to use vectors while in CICS
  - ✓ Interfaces with standard callable routines
- Perform Verb operations
  - ✓ Utilizes Vectors for comparisons, and increments
  - ✓ Operates with as much data loaded into Vectors
- High performance operation
- Maintaining maximum instructions and data on chip and not memory

# Performance Improvements

- Machine Support
  - ✓ z Series - Support for 64bit register usage
  - ✓ z10 - Decimal Floating Point
  - ✓ EC12 and BC12 - Decimal Floating Point Zone Extension
  - ✓ z13 and z13s - 32 Vector Registers and Instructions
  - ✓ z14 and z14 ZR1 - Vector Packed Decimal Facility
  - ✓ z15 and z15 T02 - Enhanced Vector Facility
  - ✓ z16 - Advanced Vector Facility Packed Decimal
- Structural Support
  - ✓ Highly optimized code generation
  - ✓ Use of registers, and vector registers, for intermediate results
  - ✓ Persistent and compressed code generation

# Cobol Bench Mark (part 1)

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  BENCH  IS INITIAL PROGRAM.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-370.
OBJECT-COMPUTER. IBM-370.

DATA DIVISION.
WORKING-STORAGE SECTION.
77   SALARY     PICTURE   9(10)V9(8)  VALUE 62.
77   TAX        PICTURE   9(10)V9(8)  VALUE 52.
77   AMOUNT     PICTURE   9(12)V9(6)  VALUE 48.
77   ONE        PICTURE   9(12)V9(6)  VALUE 1234.
77   TWO        PICTURE   9(12)V9(6)  VALUE 145.
77   THREE      PICTURE   9(12)V9(6)  VALUE 1456.
77   FOUR       PICTURE   9(12)V9(6)  VALUE 167.
77   FIVE       PICTURE   9(12)V9(6)  VALUE 1678.
77   SIX        PICTURE   9(12)V9(6)  VALUE 1789.
```

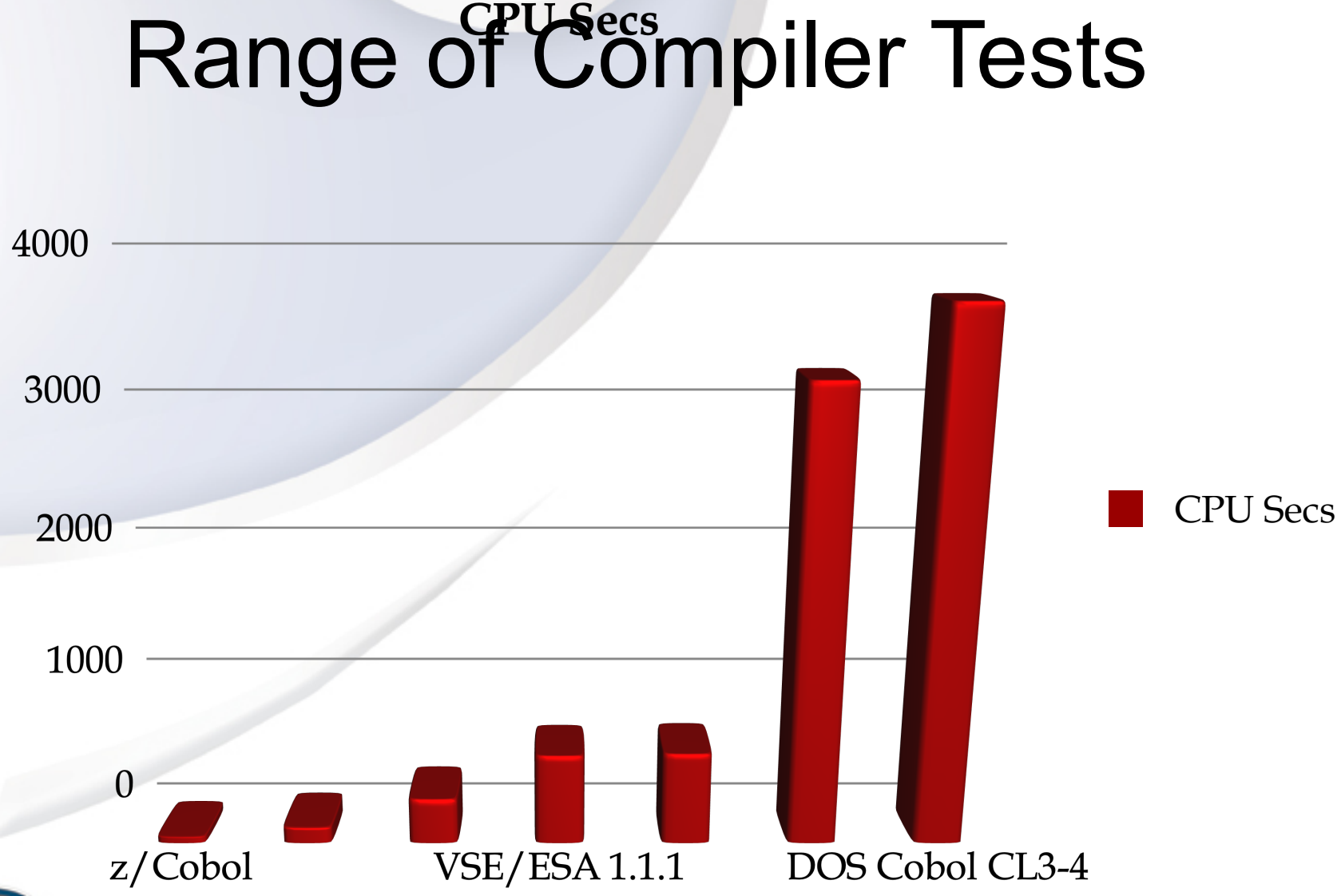
# Cobol Bench Mark (Part 2)

```
PROCEDURE DIVISION.

      PERFORM CALCULATION 100000000      TIMES.

      STOP RUN.

 CALCULATION.

      ADD ONE TWO THREE TO AMOUNT.
      SUBTRACT ONE TWO THREE FROM AMOUNT.

      MULTIPLY SALARY BY TAX.
      DIVIDE SALARY INTO TAX.

      ADD ONE TWO THREE FOUR FIVE SIX TO AMOUNT.
      SUBTRACT ONE TWO THREE FOUR FIVE SIX FROM AMOUNT.

 END PROGRAM BENCH.
```

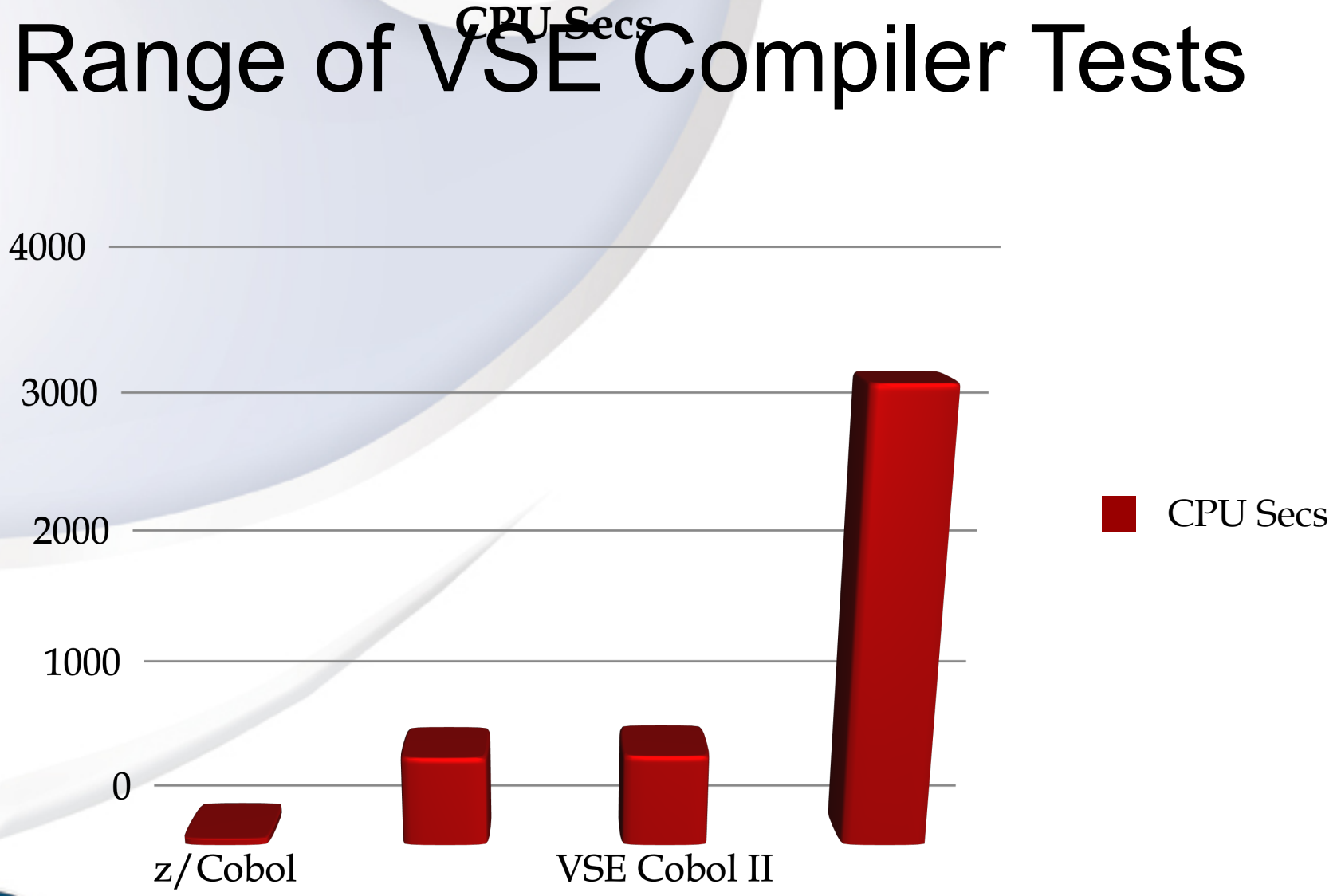
# Performance Comparison

- z/Cobol™ Code generated performs 13.74 times faster than Cobol 1.1
- Highly Math oriented Cobol program
  - ✓ Add, Subtract, Multiply, and Divide Verbs, repetitive 100,000,000 Loop
- Test on IBM Cobol for VSE/ESA 1.1.1
  - ✓ z14 ZR1 – A02 Msu 21
  - ✓ CPU Secs 646.81
- Test on z/Cobol for VSE 2.1
  - ✓ z14 ZR1 – A02 Msu 21
  - ✓ CPU Secs 047.06
- CPU improvement is 13.74 times faster in same environment

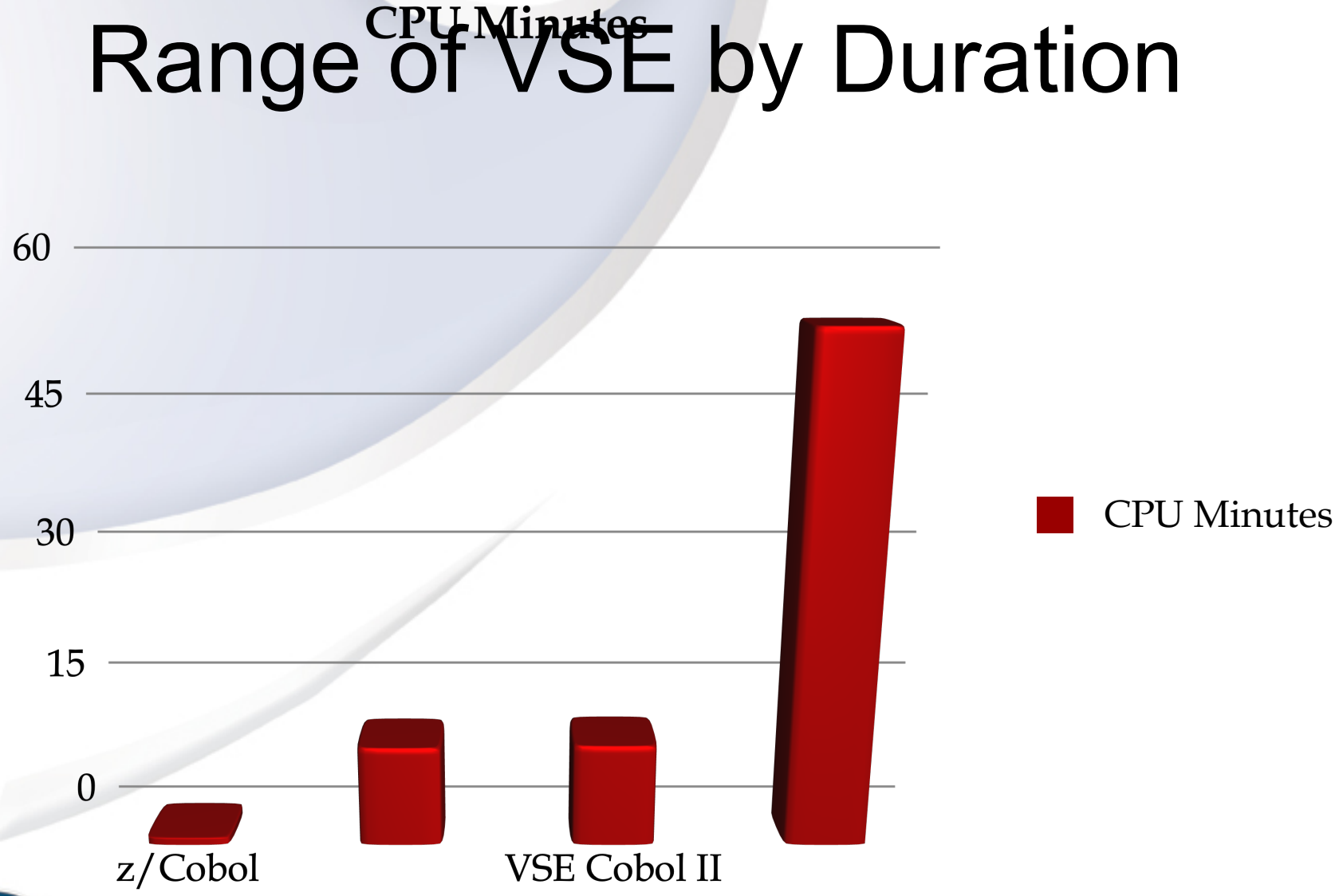
# Range of Compiler Tests



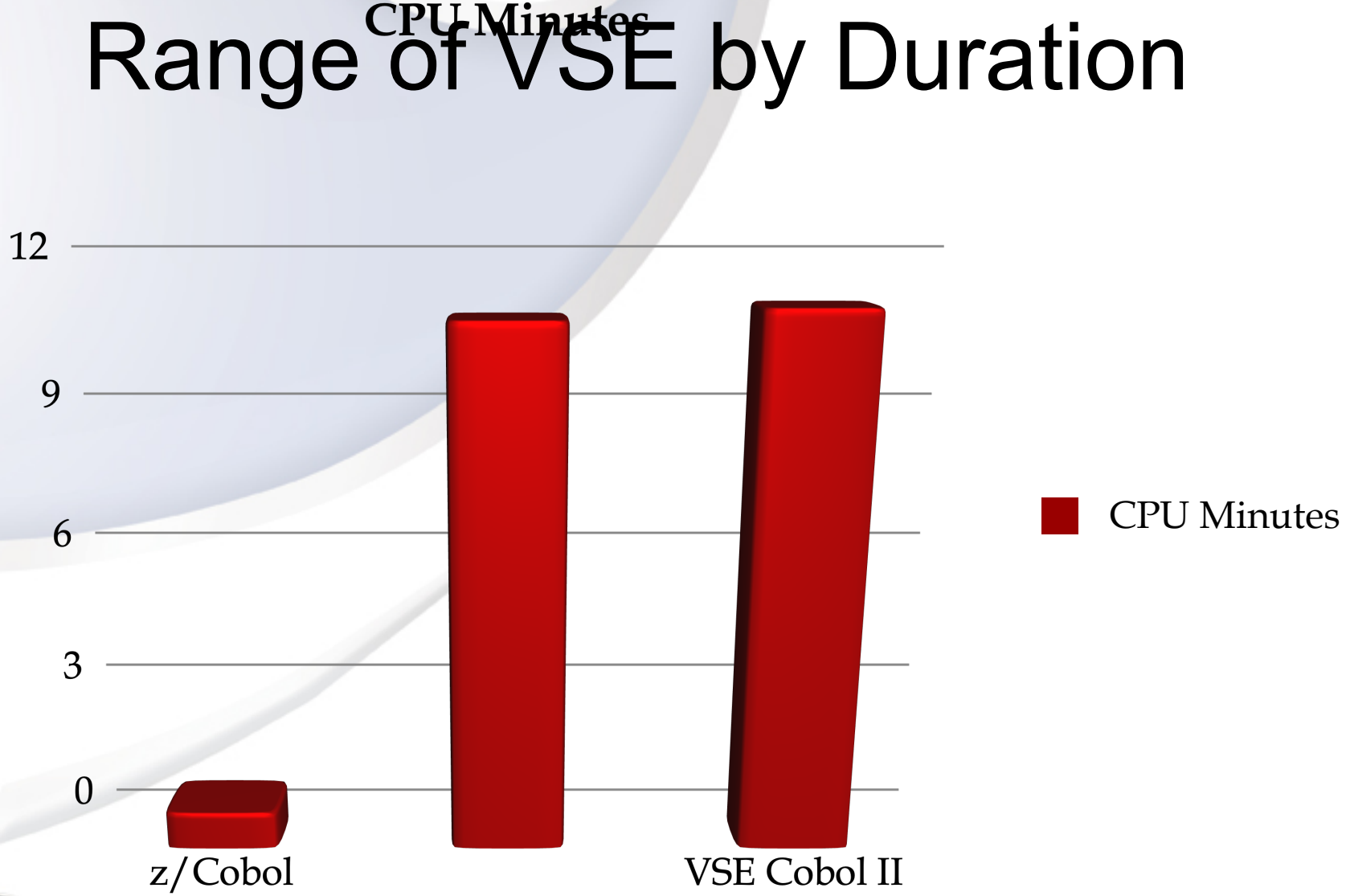
# Range of VSE Compiler Tests



# Range of VSE by Duration



# Range of VSE by Duration



# Raw Data Results

| | |
|---|---|
| • z/Cobol™ | 0047.06 |
| • Cobol VSE/ESA 1.1.1 | 0646.81 |
| • VS Cobol II 4.0 | 0660.62 |
| • DOS/VS Cobol 3.1 | 3207.89 |
| • DOS Cobol CBF CL3-4 | 3701.47 |
| • Enterprise Cobol for z/OS 6.3 (Default) | 0324.60 |
| • Enterprise Cobol for z/OS 6.3 (Arch 12) | 0115.80 |

# DOS Cobol CBF CL3-4

```
ADD ONE TWO THREE FOUR FIVE SIX TO AMOUNT.
            PACK  1C0(15,13),036(16,6)  TS=01          DNM=1-95
            PACK  1CE(2,13),045(3,6)    TS=015         DNM=1-95+15
            PACK  1D0(15,13),048(16,6)  TS=017         DNM=1-108
            PACK  1DE(2,13),057(3,6)    TS=031         DNM=1-108+15
            AP    1C6(10,13),1D6(10,13) TS=07          TS=023
            PACK  1D0(15,13),05A(16,6)  TS=017         DNM=1-121
            PACK  1DE(2,13),069(3,6)    TS=031         DNM=1-121+15
            AP    1C6(10,13),1D6(10,13) TS=07          TS=023
            PACK  1D0(15,13),06C(16,6)  TS=017         DNM=1-136
            PACK  1DE(2,13),07B(3,6)    TS=031         DNM=1-136+15
            AP    1C6(10,13),1D6(10,13) TS=07          TS=023
            PACK  1D0(15,13),07E(16,6)  TS=017         DNM=1-150
            PACK  1DE(2,13),08D(3,6)    TS=031         DNM=1-150+15
            AP    1C6(10,13),1D6(10,13) TS=07          TS=023
            PACK  1D0(15,13),090(16,6)  TS=017         DNM=1-164
            PACK  1DE(2,13),09F(3,6)    TS=031         DNM=1-164+15
            AP    1C6(10,13),1D6(10,13) TS=07          TS=023
            PACK  1D0(15,13),024(16,6)  TS=017         DNM=1-79
            PACK  1DE(2,13),033(3,6)    TS=031         DNM=1-79+15
            AP    1C6(10,13),1D6(10,13) TS=07          TS=023
            UNPK  024(16,6),1C6(9,13)   DNM=1-79       TS=07
            UNPK  033(3,6),1CE(2,13)    DNM=1-79+15    TS=07+8
            OI    035(6),X'F0'          DNM=1-79+17
```

# DOS/VSE Cobol 3.1

```
ADD ONE TWO THREE FOUR FIVE SIX TO AMOUNT.
            PACK   220(15,13),036(16,6)  TS=01          DNM=1-95
            PACK   22E(2,13),045(3,6)    TS=015         DNM=1-95+15
            PACK   230(15,13),048(16,6)  TS=017         DNM=1-108
            PACK   23E(2,13),057(3,6)    TS=031         DNM=1-108+15
            AP     226(10,13),236(10,13) TS=07          TS=023
            PACK   230(15,13),05A(16,6)  TS=017         DNM=1-121
            PACK   23E(2,13),069(3,6)    TS=031         DNM=1-121+15
            AP     226(10,13),236(10,13) TS=07          TS=023
            PACK   230(15,13),06C(16,6)  TS=017         DNM=1-136
            PACK   23E(2,13),07B(3,6)    TS=031         DNM=1-136+15
            AP     226(10,13),236(10,13) TS=07          TS=023
            PACK   230(15,13),07E(16,6)  TS=017         DNM=1-150
            PACK   23E(2,13),08D(3,6)    TS=031         DNM=1-150+15
            AP     226(10,13),236(10,13) TS=07          TS=023
            PACK   230(15,13),090(16,6)  TS=017         DNM=1-164
            PACK   23E(2,13),09F(3,6)    TS=031         DNM=1-164+15
            AP     226(10,13),236(10,13) TS=07          TS=023
            PACK   230(15,13),024(16,6)  TS=017         DNM=1-79
            PACK   23E(2,13),033(3,6)    TS=031         DNM=1-79+15
            AP     226(10,13),236(10,13) TS=07          TS=023
            UNPK   024(16,6),226(9,13)   DNM=1-79       TS=07
            UNPK   033(3,6),22E(2,13)    DNM=1-79+15    TS=07+8
            OI     035(6),X'F0'          DNM=1-79+17
```

# VS Cobol II 4.0

```
ADD ONE TWO THREE FOUR FIVE SIX TO AMOUNT.
        PACK 408(9,13),96(16,9)      TS1=0           TWO
        PACK 416(2,13),111(3,9)      TS1=8           TWO+15
        OI   417(13),X'0F'           TS1=9
        PACK 424(9,13),72(16,9)      TS2=0           ONE
        PACK 432(2,13),87(3,9)       TS2=8           ONE+15
        OI   433(13),X'0F'           TS2=9
        AP   408(10,13),424(10,13)   TS1=0           TS2=0
        PACK 424(9,13),120(16,9)     TS2=0           THREE
        PACK 432(2,13),135(3,9)      TS2=8           THREE+15
        OI   433(13),X'0F'           TS2=9
        AP   408(10,13),424(10,13)   TS1=0           TS2=0
        PACK 424(9,13),144(16,9)     TS2=0           FOUR
        PACK 432(2,13),159(3,9)      TS2=8           FOUR+15
        OI   433(13),X'0F'           TS2=9
        AP   408(10,13),424(10,13)   TS1=0           TS2=0
        PACK 424(9,13),168(16,9)     TS2=0           FIVE
        PACK 432(2,13),183(3,9)      TS2=8           FIVE+15
        OI   433(13),X'0F'           TS2=9
        AP   408(10,13),424(10,13)   TS1=0           TS2=0
        PACK 424(9,13),192(16,9)     TS2=0           SIX
        PACK 432(2,13),207(3,9)      TS2=8           SIX+15
        OI   433(13),X'0F'           TS2=9
        AP   408(10,13),424(10,13)   TS1=0           TS2=0
        PACK 424(9,13),48(16,9)      TS2=0           AMOUNT
        PACK 432(2,13),63(3,9)       TS2=8           AMOUNT+15
        OI   433(13),X'0F'           TS2=9
        AP   424(10,13),408(10,13)   TS2=0           TS1=0
        UNPK 48(4,9),424(3,13)       AMOUNT          TS2=0
        UNPK 51(15,9),426(8,13)      AMOUNT+3        TS2=2
        OI   65(9),X'F0'             AMOUNT+17
```

# Cobol VSE/ESA 1.1.1

```
ADD ONE TWO THREE FOUR FIVE SIX TO AMOUNT.
        PACK 408(3,13),96(4,9)       TS1=0          TWO
        PACK 410(8,13),99(15,9)      TS1=2          TWO+3
        OI   417(13),X'0F'           TS1=9
        PACK 424(3,13),72(4,9)       TS2=0          ONE
        PACK 426(8,13),75(15,9)      TS2=2          ONE+3
        OI   433(13),X'0F'           TS2=9
        AP   408(10,13),424(10,13)   TS1=0          TS2=0
        PACK 424(3,13),120(4,9)      TS2=0          THREE
        PACK 426(8,13),123(15,9)     TS2=2          THREE+3
        OI   433(13),X'0F'           TS2=9
        AP   408(10,13),424(10,13)   TS1=0          TS2=0
        PACK 424(3,13),144(4,9)      TS2=0          FOUR
        PACK 426(8,13),147(15,9)     TS2=2          FOUR+3
        OI   433(13),X'0F'           TS2=9
        AP   408(10,13),424(10,13)   TS1=0          TS2=0
        PACK 424(3,13),168(4,9)      TS2=0          FIVE
        PACK 426(8,13),171(15,9)     TS2=2          FIVE+3
        OI   433(13),X'0F'           TS2=9
        AP   408(10,13),424(10,13)   TS1=0          TS2=0
        PACK 424(3,13),192(4,9)      TS2=0          SIX
        PACK 426(8,13),195(15,9)     TS2=2          SIX+3
        OI   433(13),X'0F'           TS2=9
        AP   408(10,13),424(10,13)   TS1=0          TS2=0
        PACK 424(3,13),48(4,9)       TS2=0          AMOUNT
        PACK 426(8,13),51(15,9)      TS2=2          AMOUNT+3
        OI   433(13),X'0F'           TS2=9
        AP   424(10,13),408(10,13)   TS2=0          TS1=0
        UNPK 48(4,9),424(3,13)       AMOUNT         TS2=0
        UNPK 51(15,9),426(8,13)      AMOUNT+3       TS2=2
        OI   65(9),X'F0'             AMOUNT+17
```

# Enterprise Cobol 6.3

```
ADD ONE TWO THREE FOUR FIVE SIX TO AMOUNT.
          PKA      720(R13),200(18,R9)    #                AMOUNT
          PKA      648(R13),248(18,R9)    #                TWO
          PKA      664(R13),224(18,R9)    #                ONE
          AP       654(10,R13),670(10,R13) #
          PKA      704(R13),272(18,R9)    #                THREE
          AP       654(10,R13),710(10,R13) #
          PKA      632(R13),296(18,R9)    #                FOUR
          AP       654(10,R13),638(10,R13) #
          PKA      616(R13),320(18,R9)    #                FIVE
          AP       654(10,R13),622(10,R13) #
          PKA      600(R13),344(18,R9)    #                SIX
          AP       654(10,R13),606(10,R13) #
          AP       726(10,R13),654(10,R13) #
          UNPK     200(4,R9),726(3,R13)   #
          UNPK     203(15,R9),728(8,R13)  #
          OI       217(,R9),X'F0'         #
```

# Enterprise Cobol 6.3 (Arch 12)

```
ADD ONE TWO THREE FOUR FIVE SIX TO AMOUNT.
          VPKZ    VRF16,200(,R9),0x11   #  AMOUNT
          VPKZ    VRF17,248(,R9),0x11   #  TWO
          VPKZ    VRF18,224(,R9),0x11   #  ONE
          VAP     VRF17,VRF17,VRF18,0x13,12
          VPKZ    VRF18,272(,R9),0x11   #  THREE
          VAP     VRF17,VRF17,VRF18,0x13,4
          VPKZ    VRF18,296(,R9),0x11   #  FOUR
          VAP     VRF17,VRF17,VRF18,0x13,4
          VPKZ    VRF18,320(,R9),0x11   #  FIVE
          VAP     VRF17,VRF17,VRF18,0x13,4
          VPKZ    VRF18,344(,R9),0x11   #  SIX
          VAP     VRF17,VRF17,VRF18,0x13,4
          VSP     VRF16,VRF16,VRF17,0x12,10
          VUPKZ   VRF16,200(,R9),0x11   #
```

# z/Cobol™

```
ADD ONE TWO THREE FOUR FIVE SIX TO AMOUNT.
          VAP      V04,V28,V27,31,B'0000'
          VAP      V04,V04,V26,31,B'0000'
          VAP      V04,V04,V25,31,B'0000'
          VAP      V04,V04,V24,31,B'0000'
          VAP      V04,V04,V23,31,B'0000'
          VAP      V29,V04,V29,31,B'0000’
```

CSI INTERNATIONAL

# DOS Cobol CBF CL3-4

```
MULTIPLY SALARY BY TAX.
          PACK  1C0(15,13),000(16,6)  TS=01           DNM=1-50
          PACK  1CE(2,13),00F(3,6)    TS=015          DNM=1-50+15
          PACK  1D0(15,13),012(16,6)  TS=017          DNM=1-66
          PACK  1DE(2,13),021(3,6)    TS=031          DNM=1-66+15
          ZAP   060(16,13),1C6(10,13)                 TS=07
          ZAP   070(16,13),1D6(10,13)                 TS=023
          L     15,004(0,12)          V(ILBDXMU0)
          BALR  14,15
          MVO   090(16,13),090(13,13)
          MVO   083(16,13),080(16,13)
          MVO   090(16,13),090(15,13)
          ZAP   1E0(16,13),090(16,13) TS=033
          MVO   1E0(16,13),1E0(15,13) TS=033          TS=033
          MVO   1E0(16,13),1E0(15,13) TS=033          TS=033
          UNPK  012(16,6),1E1(14,13)  DNM=1-66        TS=034
          UNPK  021(3,6),1EE(2,13)    DNM=1-66+15     TS=034+13
          OI    023(6),X'F0'          DNM=1-66+17
```

# DOS/VS Cobol 3.1

```
MULTIPLY SALARY BY TAX.
          PACK   220(15,13),000(16,6)  TS=01           DNM=1-50
          PACK   22E(2,13),00F(3,6)    TS=015          DNM=1-50+15
          PACK   230(15,13),012(16,6)  TS=017          DNM=1-66
          PACK   23E(2,13),021(3,6)    TS=031          DNM=1-66+15
          ZAP    060(16,13),226(10,13)                 TS=07
          ZAP    070(16,13),236(10,13)                 TS=023
          L      15,008(0,12)          V(ILBDXMU0)
          BALR   14,15
          MVO    090(16,13),090(13,13)
          MVO    083(16,13),080(16,13)
          MVO    090(16,13),090(15,13)
          ZAP    240(16,13),090(16,13) TS=033
          SRP    240(16,13),03E(0),0   TS=033
          UNPK   012(16,6),241(14,13)  DNM=1-66        TS=034
          UNPK   021(3,6),24E(2,13)    DNM=1-66+15     TS=034+13
          OI     023(6),X'F0'          DNM=1-66+17
```

# VS Cobol II 4.0

```
MULTIPLY SALARY BY TAX.
        PACK 424(15,13),24(16,9)     TS2=0           TAX
        PACK 438(2,13),39(3,9)       TS2=14          TAX+15
        OI   439(13),X'0F'           TS2=15
        PACK 440(15,13),0(16,9)      TS2=16          SALARY
        PACK 454(2,13),15(3,9)       TS2=30          SALARY+15
        OI   455(13),X'0F'           TS2=31
        L    2,92(0,13)              TGTFIXD+92
        L    15,188(0,2)             V(IGZCXMU )
        LA   1,273(0,10)             PGMLIT AT +265
        BALR 14,15
        UNPK 24(4,9),474(3,13)       TAX             TS2=50
        UNPK 27(15,9),476(8,13)      TAX+3           TS2=52
        OI   41(9),X'F0'             TAX+17
```

# Cobol VSE/ESA 1.1.1

```
MULTIPLY SALARY BY TAX.
        PACK 424(9,13),24(4,9)       TS2=0           TAX
        PACK 432(8,13),27(15,9)      TS2=8           TAX+3
        OI   439(13),X'0F'           TS2=15
        PACK 440(9,13),0(4,9)        TS2=16          SALARY
        PACK 448(8,13),3(15,9)       TS2=24          SALARY+3
        OI   455(13),X'0F'           TS2=31
        L    2,92(0,13)              TGTFIXD+92
        L    15,188(0,2)             V(IGZCXMU )
        LA   1,289(0,10)             PGMLIT AT +281
        BALR 14,15
        UNPK 24(4,9),474(3,13)       TAX             TS2=50
        UNPK 27(15,9),476(8,13)      TAX+3           TS2=52
        OI   41(9),X'F0'             TAX+17
```

# Enterprise Cobol 6.3

```
MULTIPLY SALARY BY TAX.
        PKA     312(R13),152(18,R9)   #              SALARY
        PKA     296(R13),176(18,R9)   #              TAX
        LA      R2,328(,R13)          #  _BEtemp328
        ST      R2,392(,R13)          #
        LA      R2,312(,R13)          #  _BEtemp312
        ST      R2,396(,R13)          #
        LA      R2,296(,R13)          #  _BEtemp296
        ST      R2,400(,R13)          #
        LA      R1,392(,R13)          #  _ArgumentList2
        L       R15,120(,R3)          #
        L       R12,128(,R13)         #
        BASR    R14,R15               #  Call "IGZXXMU0"
        MVC     689(10,R13),346(R13)  #
        OI      698(,R13),X'0F'       #
        UNPK    176(4,R9),689(3,R13)  #
        UNPK    179(15,R9),691(8,R13) #
```

# Enterprise Cobol 6.3 (Arch 12)

```
MULTIPLY SALARY BY TAX.
        LA      R2,176(,R9)             #
        LA      R4,152(,R9)             #
        VPKZ    VRF16,0(,R4),0x11       #  SALARY
        VPKZ    VRF17,0(,R2),0x11       #  TAX
        VMSP    VRF16,VRF16,VRF17,0x8,14
        VPSOP   VRF16,VRF16,0x12,0x2,0
        VUPKZ   VRF16,176(,R9),0x11     #
```

# z/Cobol™

```
MULTIPLY SALARY BY TAX.
        VMP      V30,V31,V30,31,0
        VSRP     V30,V30,31,119,0
```

CSI INTERNATIONAL

# Embracing the Modernization of z/VSE® and VSE$^{n}$

- CSI International® Modernized VSE in 1995 with TCP/IP for VSE®
  - ✓ Original, and designed specifically for VSE
- z/Cobol™ is built in the model of TCP/IP for VSE®
  - ✓ Complete new build, new design, original and specifically for VSE
  - ✓ Customizable, extendable, modifiable
- Generates complete assembler code
  - ✓ Compatible with the High Level Assembler
  - ✓ LIST option displays complete assembler output during compilation
- Cobol Verb “ENTER ASSEMBLER” and “ENTER COBOL”
  - ✓ Allows for the programmer to drop into assembler and make custom code, including system macro calls.

# VSE File Elements

- ✓ Built in file support for
  - Librarian
  - BIM-Edit
  - ICCF
  - POWER
  - HFS
  - Customer extensions available
- ✓ All forms of Sequential disk and tapes supported by VSE
- ✓ All forms of VSAM files supported by VSE
- ✓ Advanced printer support for 3800s
- ✓ Complete support for IBM preprocessors

# VSE Memory Elements

- Support for three separate Working Storage Sections
  - ✓ Standard, in 31 bit above the line storage
  - ✓ Low, in 24 bit below the line storage
  - ✓ High, in 64 bit above the bar within 64bi virtual memory objects
- Support for Cobol extension LOCAL-STORAGE
  - ✓ 31 bit above the line storage
  - ✓ Reinitialized on multiple calls
  - ✓ Reentrant code produced
- Shared Working Storage Sections
  - ✓ Multiple z/Cobol™ program in the same partition
  - ✓ Cross partition sharing, with 64bit virtual memory objects

# Cobol Extensions

- Working Storage
  - ✓ ABOVE THE BAR and BELOW THE LINE
  - ✓ Allows callable routines support for all three storage areas
- Communication Section
  - ✓ Compatible with ANSI 85 Standard, and fully support TCP/IP
- File Section
  - ✓ Support Directory Structured file, not just seq., relative, and indexed.
- Report Section
  - ✓ Compatible with ANSI 85/74/68
- Vector Control Verbs for loading, unloading, storing, and saving
- Printer Control Verbs for advanced functions
- Sub Program execution on second CPU

# Compiler Extensions

- Assembler code output
  - ✓ Modifiable with macros, and direct code insertions
- Contains a full Assembler implementation
  - ✓ 64 bit code generation
  - ✓ Complete features and listing support
- File extensions
  - ✓ Customizable file routines
- Intrinsic extensions
  - ✓ 64bit z/Cobol™ Objects
  - ✓ Customer written