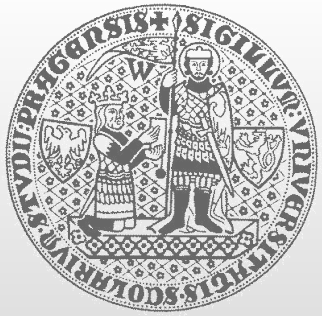
SPARC V9

Crash Dump Analysis 2015/2016



CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

Department of Distributed and Dependable Systems

ORACLE

redhat

SPARC V9 overview

- **64-bit RISC architecture**
  - Each instruction is four bytes long
  - Only load/store instructions with memory operands
  - Orthogonal instruction set
  - 32 GPRs, more in register windows
  - Big-endian

SPARC V9 overview (2)

- **64-bit RISC architecture (cont.)**
  - Branch delay slots
  - Mandatory alignment of memory accesses
  - Register windows
    - Somewhat explicit memory stack

SPARC V9 manuals

- **The SPARC Architecture Manual, Version 9**
  - http://sparc.org/wp-content/uploads/2014/01/SPARCV9.pdf.gz
- **SPARC Joint Programming Specification (JPS1): Commonality**
  - www.fujitsu.com/downloads/PRMPWR/JPS1-R1.0.4-Common-pub.pdf
- **Oracle SPARC Architecture 2015**
  - http://www.oracle.com/technetwork/server-storage/sun-sparc-enterprise/documentation/sparc-architecture-2015-2868130.pdf
- **Processor supplements**
  - http://www.oracle.com/technetwork/server-storage/sun-sparc-enterprise/documentation/index.html

SPARC V9 ABI

- **SPARC Compliance Definition 2.4.1**
  - http://sparc.org/wp-content/uploads/2014/01/SCD.2.4.1.pdf.gz
  - Authoritative source of information
  - We will use a simplified view sufficient for common cases (integer arguments, etc.)

SPARC V9 registers

- **32 64-bit GPRs**
  - r0 – r31
    - r0 reads as zero, writes ignored
    - r0 – r7 ↔ g0 – g7 (Globals)
    - r8 – r15 ↔ o0 – o7 (Outs)
    - r16 – r23 ↔ l0 – l7 (Locals)
    - r24 – r31 ↔ i0 – i7 (Ins)

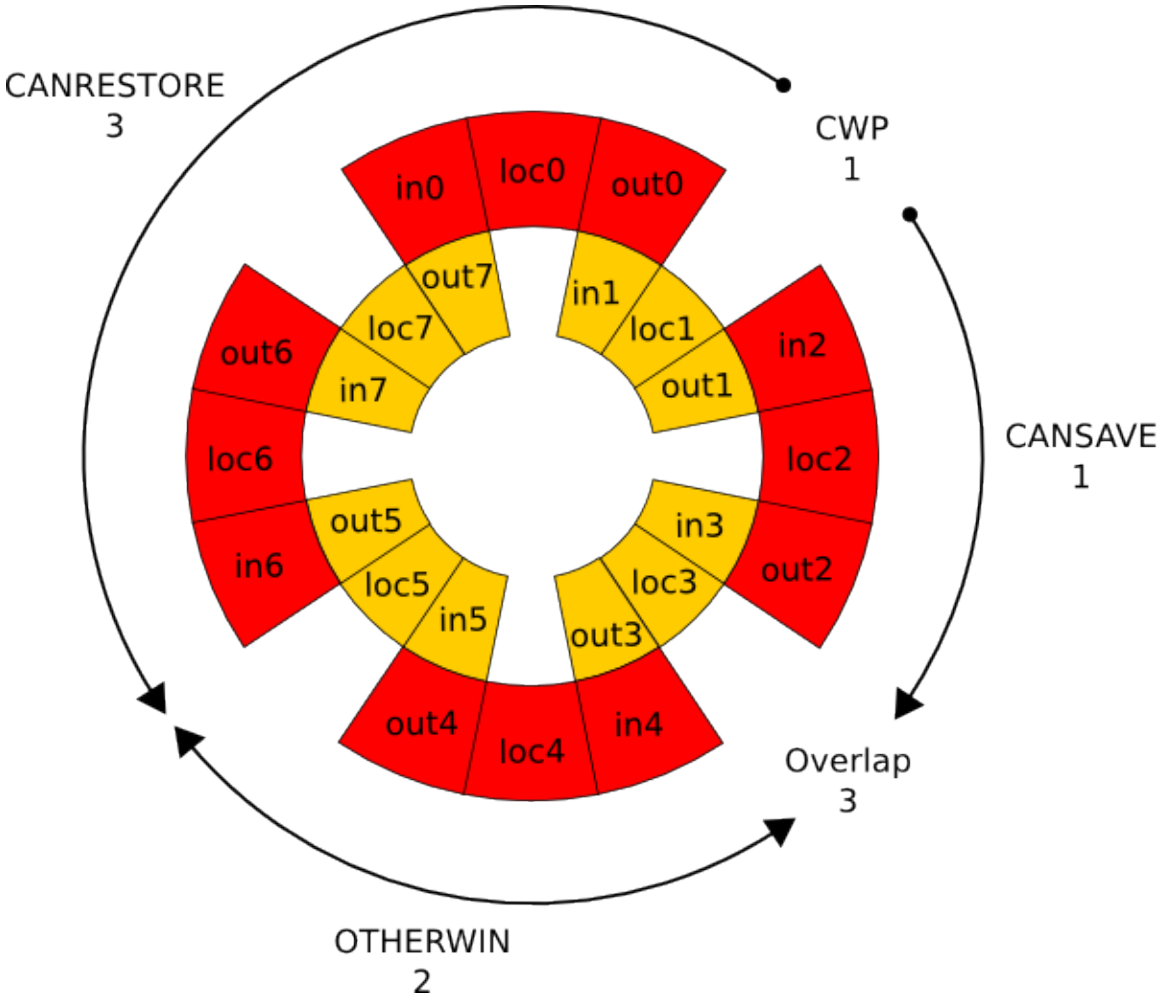
SPARC V9 registers (2)

- **Program Counter**
  - pc – current instruction
  - npc – next instruction if no trap occurs
    - Address of the target for branches
- **Integer Condition Codes Register**
  - ccr
    - icc – codes for 32-bit interpretation
    - xcc – codes for 64-bit interpretation

Register Windows

- **NWINDOW sets (8 on UltraSPARC)**
  - At any time, only one is active (current)
  - Registers r8 – r31 alias Outs, Locals and Ins of the current window
  - Window *n*'s Outs overlap with window *((n + 1) % NWINDOW)*'s Ins

Register Windows (2)



Register Windows (3)

- **CWP, CANSAVE, CANRESTORE and (OTHERWIN and CLEARWIN) are registers which define the state of the windowing mechanism**
  - Invariant:

$$NWINDOW - 2$$

$$=$$

$$CANSAVE + CANRESTORE + OTHERWIN$$

Register Windows (4)

- **CWP – Current Window Pointer**
  - Corresponds to the current function
  - CWP ← (CWP + 1) % NWINDOW
    on function call (SAVE instruction)
  - CWP ← (CWP – 1) % NWINDOW
    on function return (RESTORE instruction)

Register Windows (5)

- **CANSAVE – number of momentarily available windows for function call nesting**
  - CANSAVE ← CANSAVE – 1 on SAVE
  - CANSAVE ← CANSAVE + 1 on RESTORE
  - CANSAVE = 0 on SAVE => window spill TRAP

Register Windows (6)

- **CANRESTORE – number of momentarily available windows for function call returning**
  - CANRESTORE ← CANRESTORE – 1 on RESTORE
  - CANRESTORE ← CANRESTORE + 1 on SAVE
  - CANRESTORE = 0 on RESTORE => window fill TRAP

Register Windows (7)

- **The stack is a backing store for register windows and register windows are caching parts of the stack**
  - Window spill trap handler
    - The OS saves the window's Ins and Locals on the stack
  - Window fill trap handler
    - The OS restores the window's Ins and Locals from the stack

Flat Mode

- **It is theoretically possible to pretend there is only one register window**
  - Simpler design
  - More deterministic function duration times
  - Poorer performance
  - Up to GCC version 4.0.2
    - `gcc -mflat`

Flat Mode (2)

- **Compiler generates an alternative function prologues and epilogues**
  - No SAVE and RESTORE instructions
  - 32 GPRs registers, much like e.g. MIPS
  - We will not assume this mode

ABI in a Nutshell

- **First 6 integer arguments passed in %o0 – %o5**
  - Other or additional arguments passed on stack
- **Return value in %i0**
- **Return address in %i7**
  - But need to add 8
- **Stack pointer in %sp**
  - But need to add 2047
- **Frame pointer in %fp**
  - But need to add 2047

ABI in a Nutshell (2)

- **Stack frame needs to be 16B aligned**
- **Stack frame has a special format**
  - Window save area for Ins and Locals
  - Stack bias of 2047
    - Larger stack frames can be efficiently accessed using 13-bit signed immediate offsets in instructions

ABI in a Nutshell (3)

- **Volatile (scratch, caller-saved) registers**

  o0 – o5, o7, g1, g4 – g5

- **Non-volatile (preserved, callee-saved) registers**

  i0 – i7, l0 – l7, o6

- **Registers reserved for system**

  g6 – g7

- **Registers reserved for application**

  g2 – g3

SPARC V9 instructions

- **Only few hundreds of instructions**
  - Every instruction is 4B long, 4B-aligned
  - Variants with register or immediate operand
- **Informal classification**
  - General purpose (arithmetic, logic, branch, etc.)
  - System instructions (privileged operations)
  - FPU instructions
  - SIMD instructions (VIS I, VIS II)

SPARC V9 instructions (2)

- **Most general purpose instructions have three operands**
  - register – register – register
  - register – immediate – register
- **INST rs1, rs2, rd**
  - rd ← rs1 INST rs2
- **ADD %i0, %i1, %l3**
  - %l3 ← %i0 ADD %i1

SPARC V9 instructions (3)

- **Load / Store instructions**
  - LD [%rs1 + simm13], %rd
  - LD [%rs1 + %rs2], %rd
  - ST %rd, [%rs1 + simm13]
  - ST %rd, [%rs1 + %rs2]
  - Size suffixes (load / store instructions)
    - UB/SB (unsigned/signed byte), UH/SH (unsigned/signed halfword), UW/SW (unsigned/signed word), X (extended word)

SPARC V9 instructions (4)

- **Logical instructions**
  - Instructions with cc suffix modifies %icc and %xcc
    - Also with addition and subtraction instructions
  - Instructions with n suffix negate %rs2 before applying
- **Synthetic instructions**
  - Not real instructions
  - Understood by the assembler
  - Aliases for common uses of the real instructions

Common instructions

- CALL, JMPL, Bcccond, BRrcond, RET, RETL
- SAVE, RESTORE, RETURN, NOP
- MOV, ADD, XOR, OR, AND, ANDcc, INC, DEC, CMP, SUB, SUBcc, SLLX, SRLX
- LDX, STX, CLRX
- SETHI

Common instructions (2)

- **CALL**
  - Call function
  - Both real and synthetic instruction
    - Synthetic: JMPL address, %o7
- **JMPL address, %rd**
  - Jump and link
  - %npc ← address
  - %pc ← %pc + 4 (delay slot)

Common instructions (3)

- **Bccond**
  - (delayed) Branch on Integer Condition Code
    - Bccond{,a}{,pt|,pn} %icc, address
    - Bccond{,a}{,pt|,pn} %xcc, address
      - ccond is A (always), N (never), [N]E ([not] equal), G (greater), LE (less or equal), GE (greater or equal), L (less), etc.
      - Prediction bit
        - pn – probably not taken
        - pt – probably taken
      - Anul bit
        - a – whether or not to cancel the delay instruction

Common instructions (4)

- **BRrcond**
  - (delayed) Branch on Register Condition
    - BRrcond{,a}{,pt|,pn} %rs1, address
    - BRrcond{,a}{,pt|,pn} %rs1, address
      - rcond is [N]Z ([not] zero), LEZ (<= 0), LZ (< 0), GZ (>0), GEZ (>= 0)
      - Prediction bit
        - pn – probably not taken
        - pt – probably taken
      - Anul bit
        - a – whether or not to cancel the delay instruction

Common instructions (5)

- **RET**
  - Return from function
  - Synthetic
  - JMPL %i7+8, %g0
- **RETL**
  - Return from leaf function
  - Synthetic
  - JMPL %o7+8, %g0

Common instructions (6)

- **SAVE**
  - Allocate a new register window
    - Current Outs become new Ins
    - "ADD %rs1, imm, %rd"
      - %rs1 is from the current window
      - %rd is from the new window
      - SAVE %sp, -192, %sp
- **RESTORE**
  - Inverse operation to SAVE
  - RESTORE %rs1,imm,%rd
    - RESTORE %i0, %l1, %o0
      - "like ADD"
      - Can be used to perform last-minute arithmetics on the result
    - %sp reverted by virtue of switching to the previous window

Common instructions (7)

- **RETURN**
  - Combination of RET and RESTORE
  - Mind the delay slot
- **NOP**
  - No operation
- **MOV**
  - Move register or simm13 to register
  - Synthetic

Common instructions (8)

- **ADD, XOR, OR, AND, ANDcc, INC, DEC, CMP, SUB, SUBcc, SLLX, SRLX**
  - Add, exclusive OR, logical OR, logical AND, increment, decrement, compare, subtract, shift left logical, shift right logical
- **LDX, STX, CLRX**
  - Load from Memory, Store to Memory, Clear Memory

Common instructions (9)

- **SETHI**
  - Set high 22 bits of the source to result
    - sethi %hi(variable), %g1
    - ldx [%g1 + %lo(variable)], %g4
    - or %g1, %lo(variable), %g1

Function Prologue

```
save %sp, -imm, %sp
...
```

Function Epilogue

```
...
ret
restore R1, imm, R2
```

```
...
return %i7 + 0x8
nop
```

Stack and Code Example

- **Remember the `a()`, `b()` and `c()` from previous lessons?**
  - Compile using `gcc -O1 -m64`
  - Disassemble and single step `main()` and `a()`
  - Observe the stack

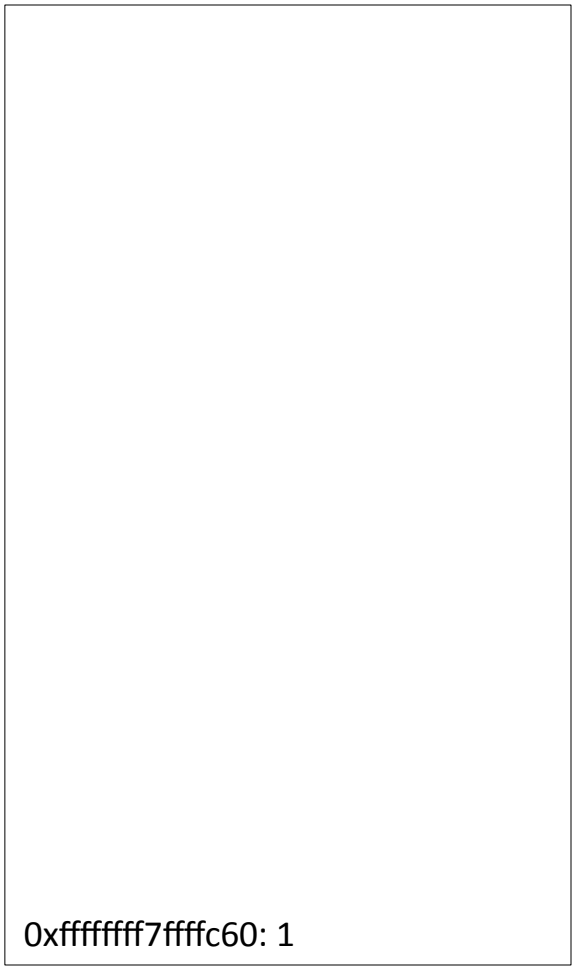
Stack and Code Example (2)

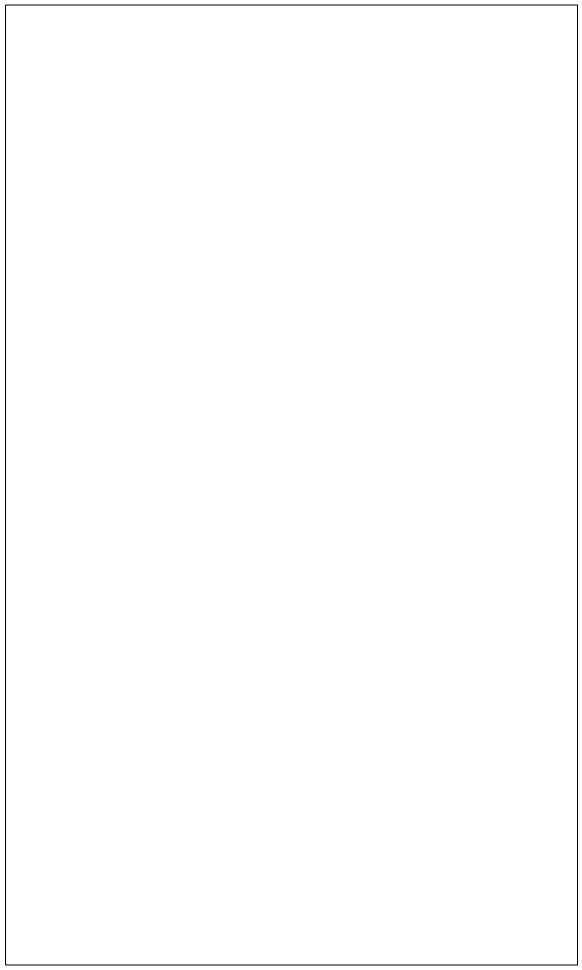
```
a:          save     %sp, -0xc0, %sp
a+4:        call     +0x10        <b>
a+8:        mov      %i0, %o0
a+0xc:      ret
a+0x10:     restore  %g0, %o0, %o0
```

```
main:       save     %sp, -0xc0, %sp
main+4:     call     -0x34        <a>
main+8:     mov      %i0, %o0
main+0xc:   ret
main+0x10:  restore  %g0, %o0, %o0
```

Stack and Code Example (2)

- **Initial state**
  - No instructions executed
  - Inherited stack pointer from main()'s caller





```
main:       save    %sp, -0xc0, %sp
main+4:     call    -0x34       <a>
main+8:     mov     %i0, %o0
main+0xc:   ret
main+0x10: restore  %g0, %o0, %o0
```

Stack and Code Example (2)

- **Allocate a new register window**
  - 192 bytes of stack space allocated
  - Old Outs became new Ins

```
0xffffffff7ffffba0: 0
0xffffffff7ffffba8: 0
0xffffffff7ffffbb0: 0
0xffffffff7ffffbb8: 0
0xffffffff7ffffbc0: 0
0xffffffff7ffffbc8: 0
0xffffffff7ffffbd0: 0
0xffffffff7ffffbd8: 0
0xffffffff7ffffbe0: 1
0xffffffff7ffffbe8: 0xffffffff7ffffd18
0xffffffff7ffffbf0: 0xffffffff7ffffd28
0xffffffff7ffffbf8: test.sparc`environ
0xffffffff7ffffc00: 0x100000000
0xffffffff7ffffc08: 0x1c00
0xffffffff7ffffc10: 0xffffffff7ffff461
0xffffffff7ffffc18: _start+0x7c
0xffffffff7ffffc20: 4
0xffffffff7ffffc28: 0xffffffff7ffffd28
0xffffffff7ffffc30: 5
0xffffffff7ffffc38: 0xffffffff7ffffda8
0xffffffff7ffffc40: 0
0xffffffff7ffffc48: 0
0xffffffff7ffffc50: 0
0xffffffff7ffffc58: 0
0xffffffff7ffffc60: 1
```

```
main:       save    %sp, -0xc0, %sp
main+4:     call    -0x34       <a>
main+8:     mov     %i0, %o0
main+0xc:   ret
main+0x10: restore  %g0, %o0, %o0
```

Stack and Code Example (2)

- **Call a( )**
  - No control transfer yet
  - %o7 ← main+4
  - %npc ← a
  - %pc ← main+8

```
main:       save   %sp, -0xc0, %sp
main+4:     call   -0x34       <a>
main+8:     mov    %i0, %o0
main+0xc:   ret
main+0x10: restore  %g0, %o0, %o0
```

```
0xffffffff7ffffba0: 0
0xffffffff7ffffba8: 0
0xffffffff7ffffbb0: 0
0xffffffff7ffffbb8: 0
0xffffffff7ffffbc0: 0
0xffffffff7ffffbc8: 0
0xffffffff7ffffbd0: 0
0xffffffff7ffffbd8: 0
0xffffffff7ffffbe0: 1
0xffffffff7ffffbe8: 0xffffffff7ffffd18
0xffffffff7ffffbf0: 0xffffffff7ffffd28
0xffffffff7ffffbf8: test.sparc`environ
0xffffffff7ffffc00: 0x100000000
0xffffffff7ffffc08: 0x1c00
0xffffffff7ffffc10: 0xffffffff7ffff461
0xffffffff7ffffc18: _start+0x7c
0xffffffff7ffffc20: 4
0xffffffff7ffffc28: 0xffffffff7ffffd28
0xffffffff7ffffc30: 5
0xffffffff7ffffc38: 0xffffffff7ffffda8
0xffffffff7ffffc40: 0
0xffffffff7ffffc48: 0
0xffffffff7ffffc50: 0
0xffffffff7ffffc58: 0
0xffffffff7ffffc60: 1
```

Stack and Code Example (2)

- **Delay slot instruction**
  - Copy incoming argument to outgoing argument

```
main:       save    %sp, -0xc0, %sp
main+4:     call    -0x34       <a>
main+8:     mov     %i0, %o0
main+0xc:   ret
main+0x10: restore  %g0, %o0, %o0
```

```
0xffffffff7ffffba0: 0
0xffffffff7ffffba8: 0
0xffffffff7ffffbb0: 0
0xffffffff7ffffbb8: 0
0xffffffff7ffffbc0: 0
0xffffffff7ffffbc8: 0
0xffffffff7ffffbd0: 0
0xffffffff7ffffbd8: 0
0xffffffff7ffffbe0: 1
0xffffffff7ffffbe8: 0xffffffff7ffffd18
0xffffffff7ffffbf0: 0xffffffff7ffffd28
0xffffffff7ffffbf8: test.sparc`environ
0xffffffff7ffffc00: 0x100000000
0xffffffff7ffffc08: 0x1c00
0xffffffff7ffffc10: 0xffffffff7ffff461
0xffffffff7ffffc18: _start+0x7c
0xffffffff7ffffc20: 4
0xffffffff7ffffc28: 0xffffffff7ffffd28
0xffffffff7ffffc30: 5
0xffffffff7ffffc38: 0xffffffff7ffffda8
0xffffffff7ffffc40: 0
0xffffffff7ffffc48: 0
0xffffffff7ffffc50: 0
0xffffffff7ffffc58: 0
0xffffffff7ffffc60: 1
```

Stack and Code Example (2)

```
a:       save    %sp, -0xc0, %sp
a+4:     call    +0x10       <b>
a+8:     mov     %i0, %o0
a+0xc:   ret
a+0x10: restore  %g0, %o0, %o0
```

- **Allocate a new register window**
  - 192 bytes of stack space allocated
  - Old Outs became new Ins

```
0xffffffff7ffffba0: 0
0xffffffff7ffffba8: 0
0xffffffff7ffffbb0: 0
0xffffffff7ffffbb8: 0
0xffffffff7ffffbc0: 0
0xffffffff7ffffbc8: 0
0xffffffff7ffffbd0: 0
0xffffffff7ffffbd8: 0
0xffffffff7ffffbe0: 1
0xffffffff7ffffbe8: 0xffffffff7ffffd18
0xffffffff7ffffbf0: 0xffffffff7ffffd28
0xffffffff7ffffbf8: test.sparc`environ
0xffffffff7ffffc00: 0x100000000
0xffffffff7ffffc08: 0x1c00
0xffffffff7ffffc10: 0xffffffff7ffff461
0xffffffff7ffffc18: _start+0x7c
0xffffffff7ffffc20: 4
0xffffffff7ffffc28: 0xffffffff7ffffd28
0xffffffff7ffffc30: 5
0xffffffff7ffffc38: 0xffffffff7ffffda8
0xffffffff7ffffc40: 0
0xffffffff7ffffc48: 0
0xffffffff7ffffc50: 0
0xffffffff7ffffc58: 0
0xffffffff7ffffc60: 1
```

```
0xffffffff7ffffae0: 0
0xffffffff7ffffae8: 0
0xffffffff7ffffaf0:  0
0xffffffff7ffffaf8:  0
0xffffffff7ffffb00: 0
0xffffffff7ffffb08: 0
0xffffffff7ffffb10: 0
0xffffffff7ffffb18: 0
0xffffffff7ffffb20: 1
0xffffffff7ffffb28: 0
0xffffffff7ffffb30: 0
0xffffffff7ffffb38: 0
0xffffffff7ffffb40: 0
0xffffffff7ffffb48: 0
0xffffffff7ffffb50: 0xffffffff7ffff3a1
0xffffffff7ffffb58: main+4
0xffffffff7ffffb60: 0
0xffffffff7ffffb68: 0
0xffffffff7ffffb70: 0
0xffffffff7ffffb78: 0
0xffffffff7ffffb80: 0
0xffffffff7ffffb88: 0
0xffffffff7ffffb90: 0
0xffffffff7ffffb98: 0xffffffff7f736c90
```

Stack and Code Example (2)

```
a:       save    %sp, -0xc0, %sp
a+4:     call    +0x10       <b>
a+8:     mov     %i0, %o0
a+0xc:   ret
a+0x10: restore  %g0, %o0, %o0
```

- **Call b( )**
  - No control transfer yet
  - %o7 ← a+4
  - %npc ← b
  - %pc ← a+8

```
0xffffffff7ffffba0: 0
0xffffffff7ffffba8: 0
0xffffffff7ffffbb0: 0
0xffffffff7ffffbb8: 0
0xffffffff7ffffbc0: 0
0xffffffff7ffffbc8: 0
0xffffffff7ffffbd0: 0
0xffffffff7ffffbd8: 0
0xffffffff7ffffbe0: 1
0xffffffff7ffffbe8: 0xffffffff7ffffd18
0xffffffff7ffffbf0: 0xffffffff7ffffd28
0xffffffff7ffffbf8: test.sparc`environ
0xffffffff7ffffc00: 0x100000000
0xffffffff7ffffc08: 0x1c00
0xffffffff7ffffc10: 0xffffffff7ffff461
0xffffffff7ffffc18: _start+0x7c
0xffffffff7ffffc20: 4
0xffffffff7ffffc28: 0xffffffff7ffffd28
0xffffffff7ffffc30: 5
0xffffffff7ffffc38: 0xffffffff7ffffda8
0xffffffff7ffffc40: 0
0xffffffff7ffffc48: 0
0xffffffff7ffffc50: 0
0xffffffff7ffffc58: 0
0xffffffff7ffffc60: 1
```

```
0xffffffff7ffffae0: 0
0xffffffff7ffffae8: 0
0xffffffff7ffffaf0:  0
0xffffffff7ffffaf8:  0
0xffffffff7ffffb00: 0
0xffffffff7ffffb08: 0
0xffffffff7ffffb10: 0
0xffffffff7ffffb18: 0
0xffffffff7ffffb20: 1
0xffffffff7ffffb28: 0
0xffffffff7ffffb30: 0
0xffffffff7ffffb38: 0
0xffffffff7ffffb40: 0
0xffffffff7ffffb48: 0
0xffffffff7ffffb50: 0xffffffff7ffff3a1
0xffffffff7ffffb58: main+4
0xffffffff7ffffb60: 0
0xffffffff7ffffb68: 0
0xffffffff7ffffb70: 0
0xffffffff7ffffb78: 0
0xffffffff7ffffb80: 0
0xffffffff7ffffb88: 0
0xffffffff7ffffb90: 0
0xffffffff7ffffb98: 0xffffffff7f736c90
```

Stack and Code Example (2)

```
a:      save    %sp, -0xc0, %sp
a+4:    call    +0x10       <b>
a+8:    mov     %i0, %o0
a+0xc:  ret
a+0x10: restore   %g0, %o0, %o0
```

- **Delay slot instruction**
  - Copy incoming argument to outgoing argument

```
0xffffffff7ffffba0: 0
0xffffffff7ffffba8: 0
0xffffffff7ffffbb0: 0
0xffffffff7ffffbb8: 0
0xffffffff7ffffbc0: 0
0xffffffff7ffffbc8: 0
0xffffffff7ffffbd0: 0
0xffffffff7ffffbd8: 0
0xffffffff7ffffbe0: 1
0xffffffff7ffffbe8: 0xffffffff7ffffd18
0xffffffff7ffffbf0: 0xffffffff7ffffd28
0xffffffff7ffffbf8: test.sparc`environ
0xffffffff7ffffc00: 0x100000000
0xffffffff7ffffc08: 0x1c00
0xffffffff7ffffc10: 0xffffffff7ffff461
0xffffffff7ffffc18: _start+0x7c
0xffffffff7ffffc20: 4
0xffffffff7ffffc28: 0xffffffff7ffffd28
0xffffffff7ffffc30: 5
0xffffffff7ffffc38: 0xffffffff7ffffda8
0xffffffff7ffffc40: 0
0xffffffff7ffffc48: 0
0xffffffff7ffffc50: 0
0xffffffff7ffffc58: 0
0xffffffff7ffffc60: 1
```

```
0xffffffff7ffffae0: 0
0xffffffff7ffffae8: 0
0xffffffff7ffffaf0:  0
0xffffffff7ffffaf8:  0
0xffffffff7ffffb00: 0
0xffffffff7ffffb08: 0
0xffffffff7ffffb10: 0
0xffffffff7ffffb18: 0
0xffffffff7ffffb20: 1
0xffffffff7ffffb28: 0
0xffffffff7ffffb30: 0
0xffffffff7ffffb38: 0
0xffffffff7ffffb40: 0
0xffffffff7ffffb48: 0
0xffffffff7ffffb50: 0xffffffff7ffff3a1
0xffffffff7ffffb58: main+4
0xffffffff7ffffb60: 0
0xffffffff7ffffb68: 0
0xffffffff7ffffb70: 0
0xffffffff7ffffb78: 0
0xffffffff7ffffb80: 0
0xffffffff7ffffb88: 0
0xffffffff7ffffb90: 0
0xffffffff7ffffb98: 0xffffffff7f736c90
```

Stack and Code Example (2)

```
a:       save    %sp, -0xc0, %sp
a+4:     call    +0x10       <b>
a+8:     mov     %i0, %o0
a+0xc:   ret
a+0x10: restore  %g0, %o0, %o0
```

- **Step through and return from `b()`**

```
0xffffffff7ffffba0: 0
0xffffffff7ffffba8: 0
0xffffffff7ffffbb0: 0
0xffffffff7ffffbb8: 0
0xffffffff7ffffbc0: 0
0xffffffff7ffffbc8: 0
0xffffffff7ffffbd0: 0
0xffffffff7ffffbd8: 0
0xffffffff7ffffbe0: 1
0xffffffff7ffffbe8: 0xffffffff7ffffd18
0xffffffff7ffffbf0: 0xffffffff7ffffd28
0xffffffff7ffffbf8: test.sparc`environ
0xffffffff7ffffc00: 0x100000000
0xffffffff7ffffc08: 0x1c00
0xffffffff7ffffc10: 0xffffffff7ffff461
0xffffffff7ffffc18: _start+0x7c
0xffffffff7ffffc20: 4
0xffffffff7ffffc28: 0xffffffff7ffffd28
0xffffffff7ffffc30: 5
0xffffffff7ffffc38: 0xffffffff7ffffda8
0xffffffff7ffffc40: 0
0xffffffff7ffffc48: 0
0xffffffff7ffffc50: 0
0xffffffff7ffffc58: 0
0xffffffff7ffffc60: 1
```

```
0xffffffff7ffffae0: 0
0xffffffff7ffffae8: 0
0xffffffff7ffffaf0:  0
0xffffffff7ffffaf8:  0
0xffffffff7ffffb00: 0
0xffffffff7ffffb08: 0
0xffffffff7ffffb10: 0
0xffffffff7ffffb18: 0
0xffffffff7ffffb20: 1
0xffffffff7ffffb28: 0
0xffffffff7ffffb30: 0
0xffffffff7ffffb38: 0
0xffffffff7ffffb40: 0
0xffffffff7ffffb48: 0
0xffffffff7ffffb50: 0xffffffff7ffff3a1
0xffffffff7ffffb58: main+4
0xffffffff7ffffb60: 0
0xffffffff7ffffb68: 0
0xffffffff7ffffb70: 0
0xffffffff7ffffb78: 0
0xffffffff7ffffb80: 0
0xffffffff7ffffb88: 0
0xffffffff7ffffb90: 0
0xffffffff7ffffb98: 0xffffffff7f736c90
```

Stack and Code Example (2)

```
a:        save    %sp, -0xc0, %sp
a+4:      call    +0x10       <b>
a+8:      mov     %i0, %o0
a+0xc:    ret
a+0x10: restore   %g0, %o0, %o0
```

- **Return from a()**
  - No control transfer yet
  - %npc ← %i7+8
  - %pc ← a+0x10

```
0xffffffff7ffffba0: 0
0xffffffff7ffffba8: 0
0xffffffff7ffffbb0: 0
0xffffffff7ffffbb8: 0
0xffffffff7ffffbc0: 0
0xffffffff7ffffbc8: 0
0xffffffff7ffffbd0: 0
0xffffffff7ffffbd8: 0
0xffffffff7ffffbe0: 1
0xffffffff7ffffbe8: 0xffffffff7ffffd18
0xffffffff7ffffbf0: 0xffffffff7ffffd28
0xffffffff7ffffbf8: test.sparc`environ
0xffffffff7ffffc00: 0x100000000
0xffffffff7ffffc08: 0x1c00
0xffffffff7ffffc10: 0xffffffff7ffff461
0xffffffff7ffffc18: _start+0x7c
0xffffffff7ffffc20: 4
0xffffffff7ffffc28: 0xffffffff7ffffd28
0xffffffff7ffffc30: 5
0xffffffff7ffffc38: 0xffffffff7ffffda8
0xffffffff7ffffc40: 0
0xffffffff7ffffc48: 0
0xffffffff7ffffc50: 0
0xffffffff7ffffc58: 0
0xffffffff7ffffc60: 1
```

```
0xffffffff7ffffae0: 0
0xffffffff7ffffae8: 0
0xffffffff7ffffaf0:  0
0xffffffff7ffffaf8:  0
0xffffffff7ffffb00: 0
0xffffffff7ffffb08: 0
0xffffffff7ffffb10: 0
0xffffffff7ffffb18: 0
0xffffffff7ffffb20: 1
0xffffffff7ffffb28: 0
0xffffffff7ffffb30: 0
0xffffffff7ffffb38: 0
0xffffffff7ffffb40: 0
0xffffffff7ffffb48: 0
0xffffffff7ffffb50: 0xffffffff7ffff3a1
0xffffffff7ffffb58: main+4
0xffffffff7ffffb60: 0
0xffffffff7ffffb68: 0
0xffffffff7ffffb70: 0
0xffffffff7ffffb78: 0
0xffffffff7ffffb80: 0
0xffffffff7ffffb88: 0
0xffffffff7ffffb90: 0
0xffffffff7ffffb98: 0xffffffff7f736c90
```

Stack and Code Example (2)

```
a:        save    %sp, -0xc0, %sp
a+4:      call    +0x10       <b>
a+8:      mov     %i0, %o0
a+0xc:    ret
a+0x10: restore   %g0, %o0, %o0
```

- **Restore the previous register window**
  - Free 192 bytes of stack space
  - Old Ins become current Outs

```
0xffffffff7ffffba0: 0
0xffffffff7ffffba8: 0
0xffffffff7ffffbb0: 0
0xffffffff7ffffbb8: 0
0xffffffff7ffffbc0: 0
0xffffffff7ffffbc8: 0
0xffffffff7ffffbd0: 0
0xffffffff7ffffbd8: 0
0xffffffff7ffffbe0: 1
0xffffffff7ffffbe8: 0xffffffff7ffffd18
0xffffffff7ffffbf0: 0xffffffff7ffffd28
0xffffffff7ffffbf8: test.sparc`environ
0xffffffff7ffffc00: 0x100000000
0xffffffff7ffffc08: 0x1c00
0xffffffff7ffffc10: 0xffffffff7ffff461
0xffffffff7ffffc18: _start+0x7c
0xffffffff7ffffc20: 4
0xffffffff7ffffc28: 0xffffffff7ffffd28
0xffffffff7ffffc30: 5
0xffffffff7ffffc38: 0xffffffff7ffffda8
0xffffffff7ffffc40: 0
0xffffffff7ffffc48: 0
0xffffffff7ffffc50: 0
0xffffffff7ffffc58: 0
0xffffffff7ffffc60: 1
```

Stack and Code Example (2)

- **Return from `main()`**
  - No control transfer yet
  - %npc ← %i7+8
  - %pc ← main+0x10

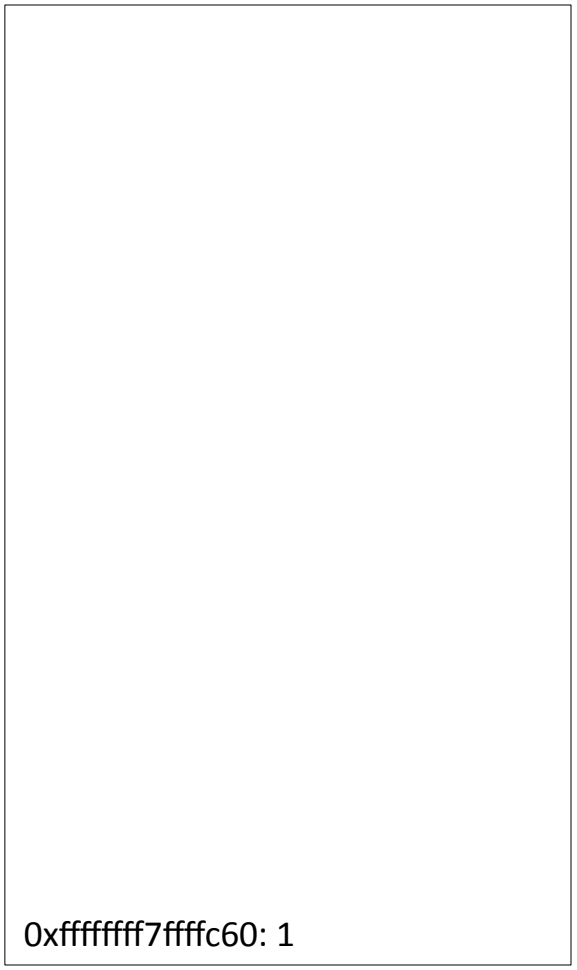
```
0xffffffff7ffffba0: 0
0xffffffff7ffffba8: 0
0xffffffff7ffffbb0: 0
0xffffffff7ffffbb8: 0
0xffffffff7ffffbc0: 0
0xffffffff7ffffbc8: 0
0xffffffff7ffffbd0: 0
0xffffffff7ffffbd8: 0
0xffffffff7ffffbe0: 1
0xffffffff7ffffbe8: 0xffffffff7ffffd18
0xffffffff7ffffbf0: 0xffffffff7ffffd28
0xffffffff7ffffbf8: test.sparc`environ
0xffffffff7ffffc00: 0x100000000
0xffffffff7ffffc08: 0x1c00
0xffffffff7ffffc10: 0xffffffff7ffff461
0xffffffff7ffffc18: _start+0x7c
0xffffffff7ffffc20: 4
0xffffffff7ffffc28: 0xffffffff7ffffd28
0xffffffff7ffffc30: 5
0xffffffff7ffffc38: 0xffffffff7ffffda8
0xffffffff7ffffc40: 0
0xffffffff7ffffc48: 0
0xffffffff7ffffc50: 0
0xffffffff7ffffc58: 0
0xffffffff7ffffc60: 1
```

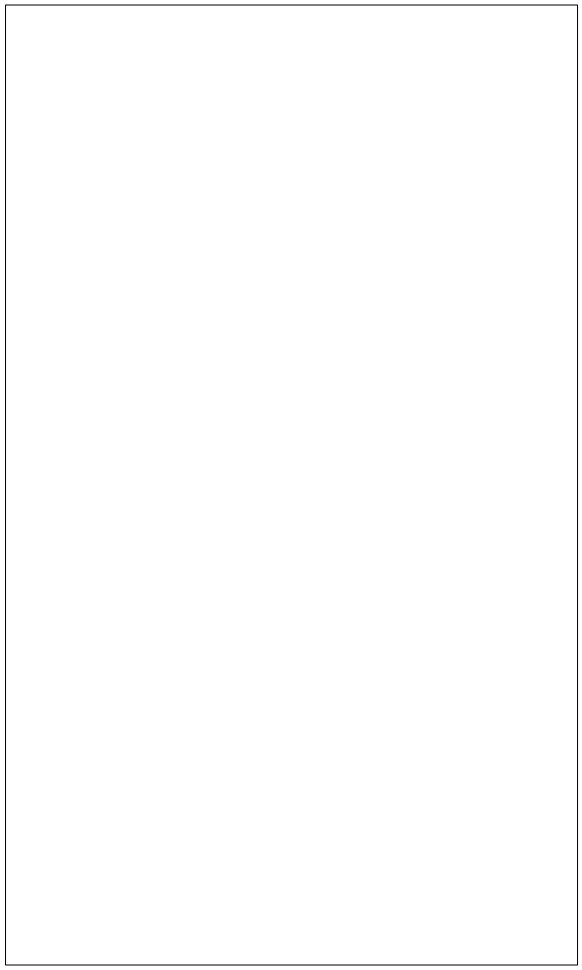
```
main:       save   %sp, -0xc0, %sp
main+4:     call   -0x34       <a>
main+8:     mov    %i0, %o0
main+0xc:   ret
main+0x10: restore  %g0, %o0, %o0
```

Stack and Code Example (2)

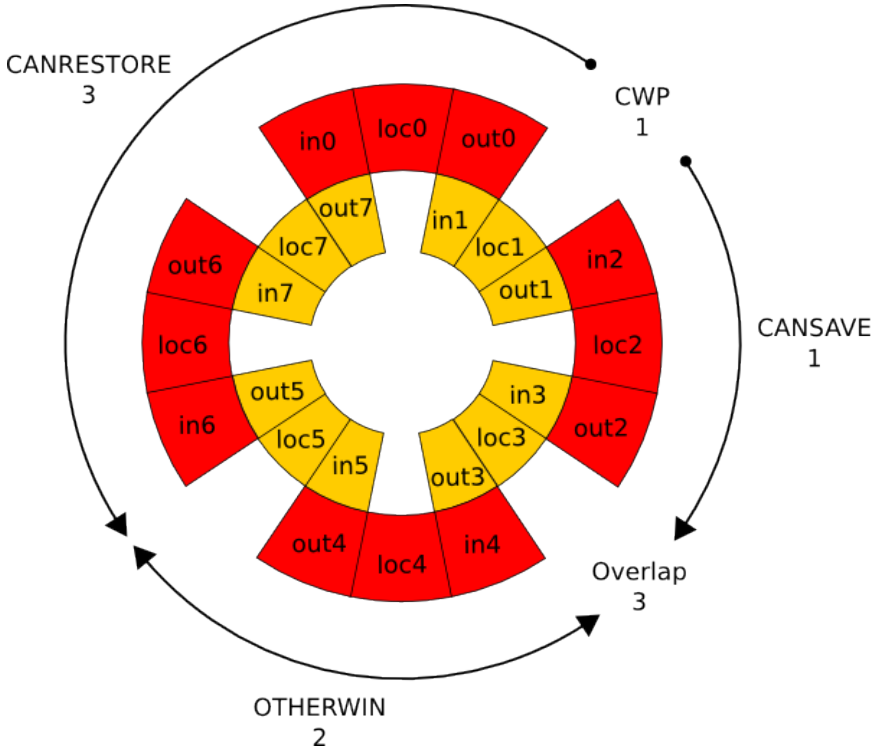
- **Restore the previous register window**
  - Free 192 bytes of stack space
  - Old Ins become current Outs

```
main:       save    %sp, -0xc0, %sp
main+4:     call    -0x34       <a>
main+8:     mov     %i0, %o0
main+0xc:   ret
main+0x10: restore  %g0, %o0, %o0
```





SPARC V9 ABI cheat sheet



i0	1st argument / ret. val
i1	2nd argument
i2	3rd argument
i3	4th argument
i4	5th argument
i5	6th argument
i6/fp	frame pointer
i7	return addr – 8

o0	1st argument for callee
o1	2nd argument for callee
o2	3rd argument for callee
o3	4th argument for callee
o4	5th argument for callee
o5	6th argument for callee
o6/sp	stack pointer
o7	where callee will return – 8

l0	
l1	
l2	
l3	
l4	
l5	
l6	
l7	

g0	always 0
g1	
g2	
g3	
g4	
g5	
g6	
g7	cur. thread In Solaris kernel

non-volatile registers
volatile registers

SPARC V9 ABI cheat sheet (2)