Arquitetura Solaris

Fernanda Santoro Jannuzzi fernanda@cbpf.br
Fernando Spencer spencer@cbpf.br
Marita Maestrelli marita@cbpf.br

Maio 2003

PREFÁCIO

As estações de trabalho da SUN são bastante versáteis e amplamente difundidas no mercado de workstations.

Nos centros de pesquisa essas estações são muito utilizadas em cálculo numérico devido a sua boa performance e a grande estabilidade de funcionamento.

No CBPF existem , hoje, 80 estações SUN entre servidoras e clientes, sendo uma com 10(dez) processadores que trabalham em paralelo, aumentado o potencial de processamento.

Esta nota técnica descreve a arquitetura das Sparc stations SUN.

ÍNDICE

Introdução

Em um sistema de computação, a destinação final do conteúdo de qualquer tipo de memória é o processador, também conhecido como CPU (central processor unit). Isto é, o objetivo final de cada uma das memórias (ou do sistema de memória) é armazenar informações destinadas a serem, em algum momento, utilizadas pelo processador. Ele é o responsável pela execução das instruções, pela manipulação dos dados e pela produção dos resultados das operações.

SPARC (Scalable Processor ARChitecture) é uma arquitetura de processador aberta, baseada em RISC (Reduced Instruction Set Computer), inclui processadores de diversas empresas e estão presentes em dispositivos que vão desde câmeras digitais a grandes servidores UNIX. É uma arquitetura controlada e gerenciada por uma fundação internacional não governamental fundada em 1989. Sua filosofia, desde o nascimento, é de um padrão aberto para promover inovação, flexibilidade e encorajar a competição. SPARC foi criada nos laboratórios da Sun Microsystems Inc, baseado na pioneira pesquisa da Universidade da Califórnia em Berkeley sobre arquitetura RISC. O primeiro produto baseado na arquitetura SPARC foi produzido pela Sun e Fujitso em 1986 e em 1987 a Sun lançou a primeira *workstation* baseada em um processador SPARC. Entre os diversos produtos da arquitetura SPARC podemos citar Computação Militar, Fotografia Digital, Computação Aeroespacial, Imagens Médicas, Computação Empresarial, Notebooks, Telecomunicações, Armazenamento Empresarial, Computadores Pessoais, Rede, Eletro-Eletrônicos para Casa.

As ações operativas do processador são realizadas nas suas unidades funcionais: na unidade aritmética e lógica - ULA (Aritmetic and Logic Unit), na unidade de ponto flutuante - UFP (Float Point Unit - FPU) ou talvez em uma unidade de processamento vetorial. No entanto, antes que a instrução seja interpretada e as unidades da CPU sejam acionadas, o processador necessita buscar a instrução de onde ela estiver armazenada (memória cache ou principal) e armazená-la em seu próprio interior, em um dispositivo de memória denominado registrador de instrução. Em seguida a este armazenamento da instrução, o processador deverá, na maioria das vezes, buscar dados da memória (cache, principal ou mesmo de unidades de disco em fita) para serem manipulados na ULA. Esses dados também precisam ser armazenados em algum local da CPU até serem efetivamente utilizados. Os resultados de um processamento (de uma soma, subtração, operação lógica, etc.) também precisam, às vezes, ser guardados temporariamente na CPU,

ou para serem novamente manipulados na ULA por uma outra instrução, ou para serem transferidos para uma memória externa à CPU. Esses dados são armazenados na CPU em pequenas unidades de memória denominadas registradores.

1. Conjunto de Instruções (Instructions Set)

Instrução é um termo utilizado para designar os comandos que são dados ao computador para que ele execute uma operação. Esses comandos devem ser enviados em uma linguagem conhecida pela máquina. Essa tarefa é realizada pelo compilador que faz a tradução de um programa escrito por um usuário para um programa em linguagem Assembly. Em seguida um assembler faz a tradução para a linguagem de máquina.

Um conjunto de instruções é conhecido como *instruction set*. As instruções da arquitetura SPARC possuem várias categorias.

1.1 Transferência de Dados e Acesso à Memória

As únicas instruções que acessam a memoria sao *load, store, prefetch, load store unsigned byte*, e as operações atômicas *swap, e compare e swap*.

Todas as instruções exceto Compare e Swap usam dois registradores r ou um registrador r e um valor imediato 13-bits (simm13) para calcular um endereço de memória de 64 bits alinhado. A IU (Integer Unit) associa um identificador ASI (address space identifier) a esse endereço. Compare e Swap usam um único registrador r para especificar um endereço de memória de 64 bits. O campo de destino de uma instrução especifica o registrador r ou f que fornece o dado para armazenamento (*store*) ou recebe o dado (*load* ou LDSTUB). Para *swap* o campo destino de destino identifica o registrador r a ser trocado atomicamente com a localização de memória endereçável. Para *compare* e *swap* o valor de um registrador r é comparado com um valor em memória do endereço calculado. Se os valores são iguais o campo destino especifica o registrador r que será trocado atomicamente com a localização de memória endereçável. Se os valores são diferentes o campo destino especifica o registrador r que recebera o valor da localização de memória endereçável, neste caso a localização permanece inalterada. O campo destino de uma instrução *prefetch* é usado para codificar o tipo de *prefetch*. Instruções inteiras *load* e *store* suportam acessos a um byte, meia-palavra, uma palavra e uma palavra-dupla. Instruções ponto-flutuante *load* e *store* suportam acessos a uma palavra, uma palavra-dupla e uma palavra quádruplo. LDSTUB acessa bytes, SWAP acessa palavras e CAS acessa palavras ou palavras-duplas. *Prefetch* acessa no mínimo 64 bytes.

As operações atômicas são operações especiais usadas para sincronização e atualização da memória pelos processos concorrentes.

1.1.1 Load para ponto-flutuante

A instrução *load simples* (LDF) copia uma palavra da memória para *f*[rd]. A instrução load palavra-dupla (LDDF) copia uma palavra dupla alinhada da memória em um registrador de precisão dupla ponto-flutuante.

1.1.2 Load para Inteiro

A instrução *load* copia um byte, uma meia-palavra, uma palavra, uma palavra extendida ou uma palavra-dupla da memória para *r*[rd] exceto a instrução LDD.

1.1.3 Compare e Swap

Estas instruções são usadas para sincronização e atualização da memória pelos processos concorrentes. A instrução CASXA compara o valor no registrador r[rs2] com a palavra-dupla em memória apontada pelo endereço em r[rs1]. Se os valores são iguais o valor em r[rd] é trocado pelo o que está contido em r[rs1]. Senão r[rd] recebe o valor de r[rs1] mas a localização em memória permanece inalterada. A instrução CASA compara os 32 bits de baixa ordem do registrador r[rs2] com a palavra em memória apontada pelo endereço em r[rs1]. Se os valores forem iguais os 32 bits de baixa ordem do registrador r[rd] são trocados com o conteudo da memória apontado pelo endereço em r[rs1] e os 32 bits de alta ordem do registrador r[rd] são setados com zero. Se os valores não forem iguais a localização em memória permance inalterada, mas o conteúdo zero-estendido da palavra em memória apontada por r[rs1] substitui os 32 bits de baixa ordem de r[rd] e os 32 bits de alta ordem do registrador r[rd] são setados com zero.

1.2 Aritmética

As instruções aritméticas *add*, *subtract*, *divide* e *multiply* utilizam 2 operandos de fonte e o resultado é escrito em um registrador de destino. As instruções aritméticas são geralmente instruções que computam um resultado em função de dois operandos. Eles tanto escrevem o resultado em um registrador *r*[rd] destino quanto o descartam. Um dos operandos de entrada é sempre *r*[rs1]. O outro operando de entrada depende do bit *i* na instrução: se *i*=0 o operando é

r[rs2], se *i*=1 o operando é a constante *simm10*, *simm11* ou *simm13*. Note que o valor de *r*[0] sempre é lido como zero, e escritas a ele são ignoradas.

1.2.1 Add

A instrução add realiza a soma entre dois registradores *r*[rs1] + *r*[rs2] se *i*=0, ou r[rs1]+sign_ext(simm13) se *i*=1, e escreve a soma em *r*[rd].

1.2.2 Subtract

A instrução *subtract* computa a subtração entre r[rs1]-r[rs2] se *i*=0, ou r[rs1]- sign_ext(simm13) se *i*=1, e escreve a diferença em *r*[rd].

1.2.3 Divide

A instrução *divide* realiza uma divisão 64-bit por 32-bit produzindo um resultado 32-bit. Se o overflow não ocorre os 32 bits menos significativos do quociente inteiro são estendidos com sinal ou com zero para 64-bits e escritos em *r*[rd].

1.2.4 Multiply

A instrução multipy realiza a multiplicação de 32-bits por 32-bits produzindo resultados de 64-bits. Ela computa *r*[rs1]<31:0> x *r*[rs2]<31:0> se *i*=0, ou *r*[rs1]<31:0> x sign_ext(simm13)<31:0> se *i*=1, escrevendo os 32 mais significantes bits do produto no registrador Y e todos os 64 bits do produto em *r*[rd].

1.3 Transferência de Controle

As instruções de transferência de controle incluem *branches*, *calls*, *jumps* e desvios condicionais (*traps*). O bit *a* anula a execução da instrução seguinte se o desvio é condicional e não foi tomado, ou se ele não é condicional e foi tomado.

1.3.1 Branch Condicional

Um branch condicional, ou seja, um desvio condicional é tomado se a condição é verdadeira. A instrução branch examina todos os 64 bits de *r*[rs1] de acordo com o campo ***rcond*** da instrução, e produz os resultados Verdadeiro ou Falso. Se verdadeiro, o desvio é tomado ou seja a instrução causa a transferência de controle para o endereço

PC+(4xsign_ext(d16hi d16lo)). Se falso o desvio não é tomado. Os branches condicionais podem ser cinco: Bicc (branch on integer condition codes), BPr (branch on integer register with predition), BPcc (branch on integer condition codes with prediction), FBfcc(branch on floating-point condition codes), FBPfcc (branch on floating-point condition codes with prediction). Estão descritos abaixo.

Bicc
op rd op2 disp22
31 30 29 25 24 22 21 0

FBfcc:
op a cond op2 disp22
31 30 29 28 25 24 22 21 0

FBPfcc:
op a cond op2 cc1 cc0 p disp19
31 30 29 28 25 24 22 21 20 19 18 0

BPr:
op a rcond op2 d16hi p rs1 d16lo
31 30 29 28 25 24 22 21 20 19 18 14 13 0

1.3.2 Branch Incondicional

Um branch incondicional, ou seja, um desvio incondicional transfere o controle incondicionalmente se sua condição especificada é setada como 0, senão a instrução nunca é executada e tem seu bit setado como 1.

1.3.3 Call and Link (CALL)

A instrução Call and Link causa uma transferência de controle incondicional do PC para o endereço PC+(4xsign_ext(disp30)).

1.3.4 Jump and Link (JMPL, RETURN)

A instrução *jump and link* (JMPL) causa a transferência de controle para o endereço dado por r[rs1]+r[rs2] se *i*=0, ou r[rs1]+sign_ext(simm13) se *i*=1.

1.3.5 Return

A instrução *return* causa a transferência de controle para o endereço alvo. O endereço alvo é r[rs1]+r[rs2] se *i*=0, ou r[rs1]+sign_ext(simm13) se *i*=1.

1.3.6 Trap (Tcc)

A instrução *trap* inicia uma ação em resposta a presença de uma exceção ou uma interrupção se a condição especificada pelo seu campo *cond* é casada com o estado corrente do registrador de códigos de condição especificado em seu campo *cc*, senão ele executa um NOP. Se a ação é tomada ele incrementa o registrador TL, computa o tipo da *trap* que é armazenada em TT[TL] e transfere para um endereço computado na tabela de *traps* apontado pela TBA (Trap Base Address). A instrução Tcc pode especificar uma de 128 tipos de trap de software. Quando uma Tcc é tomada, o valor 256 mais os sete menos significativos bits da soma do operando de entrada de Tcc é escrito em TT[TL]. A única diferença visivel entre uma trap gerada por software via uma instrução Tcc e uma trap por hardware é o número de traps no registrador TT.

1.4 Operações Lógicas

Essas instruções implementam operações lógicas em bits. Elas computam *r*[rs1] op *r*[rs2] se *i*=0, ou *r*[rs1] op sign_ext(simm13) se *i*=1.

1.4.1 Shift

Opcode	op3	x	Operation
SLL	100101	0	Shift Left Logical - 32 Bits
SRL	100110	0	Shift Right Logical - 32 Bits
SRA	100111	0	Shift Right Arithmetic - 32 Bits
SLLX	100101	1	Shift Left Logical - 64 Bits
SRLX	100110	1	Shift Right Logical - 64 Bits
SRAX	100111	1	Shift Right Arithmetic - 64 Bits

1.5 Acesso a registrador de estados

As instruções do registrador de estado lêem e escrevem o conteúdo dos registradores de estado visíveis ao software não privilegiado

(Y,CCR,ASI,PC,TICK e FPRS) e ao software privilegiado (TPC, TNPC, TSTATE, TT, TICK, TBA, PSTATE, TL, PIL, CWP, CANSAVE, CANSTORE, CLEANWIN, OTHERWIN, WSTATE, FPQ e VER).

1.6 Move condicional

As instruções de move condicional copiam um valor do registrador fonte a um registrador destino. Essas instruções aumentam a performance do processador por reduzirem o número de branches.

1.7 Gerenciamento do registador de janelas

As instruções SAVE, RESTORE e FLUSHW são usadas para gerenciar o registrador de janelas. SAVE e RESTORE são instruções não-privilegiadas e causam a inserção (*push*) ou a remoção (*pop*), respectivamente, da janela de registradores. FLUSHHW causa a remoção das janelas, exceto a janela corrente da memória.

1.8 Específicas

1.8.1 No Operation

A instrução *no operation* (NOP) não muda o estado do programa visível. NOP é um caso especial da instrução SETHI, com *imm22*=0 e *rd*=0.

1.8.2 Population Count

A instrução *population count* conta o número de bits 1 em r[rs2] se *i*=0, ou o número de bits 1 em sign_ext(simm13) se *i*=1, e armazena a conta em *r*[rd].

1.8.3 Prefetch Data

A execução de uma instrução prefetch inicia o movimento dos dados ou uma preparação futura para o movimento ou mapeamento de endereço, para reduzir a latência de *loads* e *stores* subseqüentes.

1.8.4 SETHI

A instrução *SETHI* (Set High 22 bits of Low Word) zera os 10 bits menos significativos e os 32 bits mais significativos de *r*[rd], e substitui os bits de 31 a 10 de *r*[rd] com o valor vindo do campo *imm32*. Ela é usada para construir constantes em registradores.

1.9 Sincronização de Memória

Duas formas de instruções de bloqueio de memória (MEMBAR) permitem os programas gerenciarem a ordem e a completude de referências à memória. Ordenar MEMBARs induz a ordenação parcial entre conjuntos de loads e stores e futuros loads e stores. Sequenciar MEMBARs emprega controle explícito sobre a completude de loads e stores. Ambas formas de bloqueio são codificadas em uma única instrução, com sub-funções bit-codificadas em um campo imediato.

2. Formato dos Dados

Os dados podem ser representados numericamente em qualquer base. Os humanos, por exemplo, se acostumaram a representá-los na base 10 porque têm os dedos da mão para lhe auxiliarem a contagem. Já as máquinas utilizam a representação binária, ou seja, números na base 2, porque seu funcionamento esta intimamente ligado à séries de sinais elétricos altos e baixos. Esses sinais representam apenas dois estados: ligado/desligado, verdadeiro/falso, alto/baixo, 1/0.

a) Com sinal (*Signed*)

Os números sinalizados indicam o resultado de uma operação entre números de mesma grandeza ou grandezas diferentes. Neste caso é necessária a utilização de uma notação que registre o resultado de, por exemplo, uma subtração entre um número maior e um menor. A notação utilizada é número positivo e número negativo, representados simbolicamente por + e -, respectivamente.

A distinção entre positivo e negativo é feita através da convenção conhecida como *complemento de 2*, sendo que o número positivo é representado por 0 e o número negativo é representado por 1.

b) Sem sinal (*Unsigned*)

Os números não sinalizados são aqueles números considerados apenas positivos. Na arquitetura SPARC, sejam valores inteiros, strings, booleanos e outros valores representáveis de maneira binária são armazenados como inteiros sem sinal.

Os métodos *big-endian* e *little-endian* existem como opção de distribuição do valor de um número entre os bytes. No método *little-*

endian o valor mais significativo é colocado nos bytes de menor numeração. Já no método *big-endian* o valor mais significativo é colocado nos bytes de maior numeração, é também conhecido como convenção left-right. Por exemplo, o número 104 em little-endian é apresentado na ordem 401 e em big-endian na ordem 104. A vantagem do método little-endian é que a adição e tração é feita considerando os dígitos na ordem em que foram escritos. O *big-endian* requer a leitura de trás para frente.

A arquitetura SPARC utiliza a nomenclatura *big-endian,* conforme mostra a Figura 1 abaixo.

2.1 Tipos

a) Inteiros

Eles podem ter 8, 16, 32 e 64 bits de tamanho, e podem ter precisão simples, dupla e estendida.

b) Ponto-flutuante

Os dados de tipo ponto-flutuante podem ter 32, 64 e 128 bits de tamanho, e podem, assim como os inteiros ter precisão simples, dupla e quádrupla.

2.2 Largura dos tipos de dados

Byte:

Um byte é acessado em uma instrução load/stores cuja largura um byte. Uma instrução load/store de acessa o byte endereçado nos modos little e big-endian.

Halfword:

Dois bytes são acessados em uma instrução load/stores cuja largura seja meia-palavra. O byte mais significativo (bits 15..8) é acessado no endereço especificado na instrução, o byte menos significativo (bits 7..0) é acessado no endereço+1.

Word:

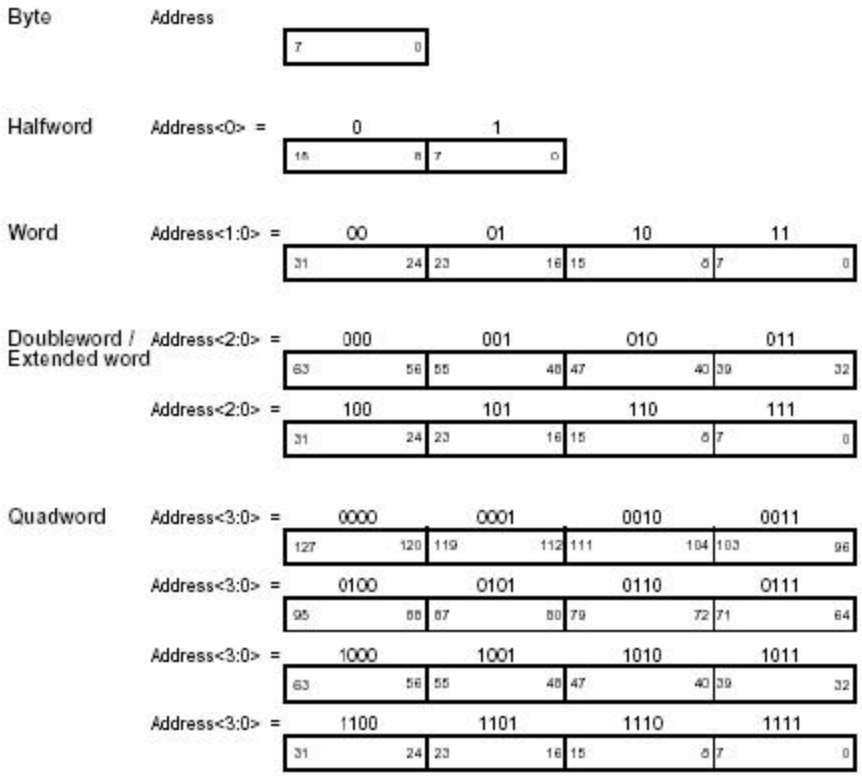
Quatro bytes sao acessados em uma instrução load/store cuja largura seja uma palavra. O byte mais significativo (bits 31..24) é acessado no

endereço especificado na instrução, o byte menos significativo (bits 7..0) é acessado no endereço+3.

doubleword or extended word:

Oito bytes são acessados em uma instrução load/stores cuja largura seja uma palavra-dupla ou estendida. O byte mais significativo (bits 63..56) é acessado no endereço especificado na instrução, o byte menos significativo (bits 7..0) é acessado no endereço+7.

Figura 1
Convenção de Endereçamento "Big-endian"



2.3 Registradores

Registrador é um dispositivo de armazenamento, como uma memória, mas que possui maior velocidade de transferência dentro do sistema (menor tempo de acesso), menor capacidade de armazenamento e maior custo.

A arquitetura SPARC possui dois tipos de registradores:

a) Registrados de propósitos gerais

Esses registradores podem ser registradores *r* para inteiro e registradores *f* para ponto flutuante. Uma unidade de inteiros (IU) pode conter de 64 a 528 registradores *r* 64-bits de propósito geral. Eles são particionados em 8 registradores globais, 8 registradores globais alternativos e 3 ou mais conjuntos que contém 16 registradores cada.

Esses conjuntos de registradores também são chamados de **janelas de registradores** e podem chegar até 32 janelas. Esses conjuntos de registradores (janelas) possuem a forma de uma pilha circular, e portanto, formam uma vizinhança entre si. Acessar uma janela implica em poder trabalhar com 24 registradores, sendo que 8 são de entrada, 8 são locais e 8 são os registradores adjacentes à janela conhecidos como de saída, conforme mostra figura abaixo.

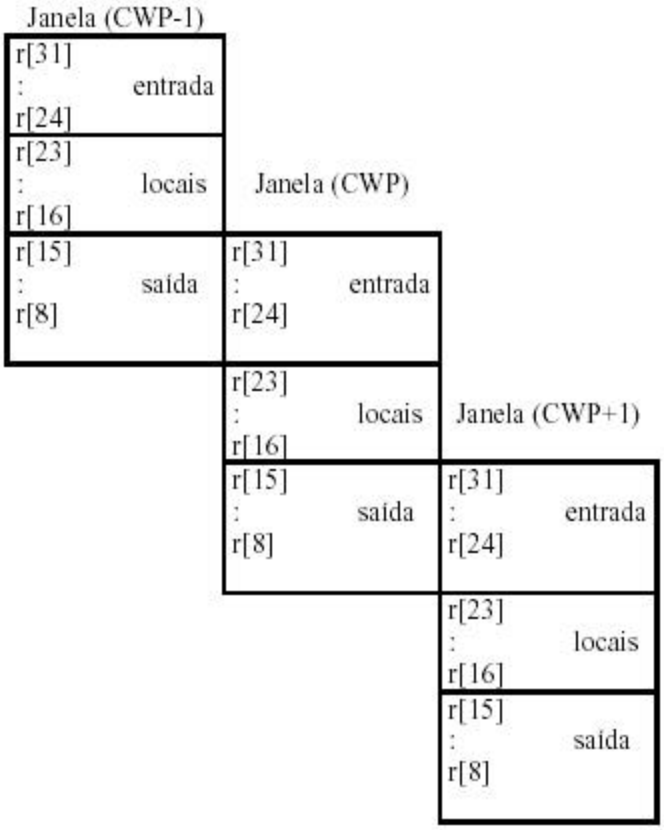


Figura 2 - Janela de Registradores (Nwindows): 16 registradores da janela (8 de entrada e 8 locais) e 8 registradores de saída formados pela vizinhança.

Assim, uma instrução pode acessar no mínimo 64 registradores *r*, sendo que 8 são registradores globais *r*, mais 8 registradores globais alternativos, mais 3 conjuntos de 16 registradores. O número máximo de registradores é 528 (8 globais, 8 globais alternativos e 32 conjuntos de 16).

Figura 3 - Registradores Globais

Cada janela (FIG.2) compartilha suas entradas com uma janela adjacente e suas saídas com outra. A saída da janela CWP-1 (módulo NWindows) é endereçável como a entrada da janela corrente, e a saída da janela corrente como a entrada da janela CWP+1 (módulo NWindows). O local é único para cada janela. Um registrador *r* com endereço *o*, onde $8 \leq o \leq 15$ se refere exatamente ao mesmo registrador como (*o*+16) faz depois de CWP ser incrementado por 1. Ainda, um registrador com endereço i, onde $24 \leq i \leq 31$ se refere exatamente ao mesmo registrador como endereço (*i*-16) faz depois que CWP é decrementado por 1, conforme mostra a figura 3.

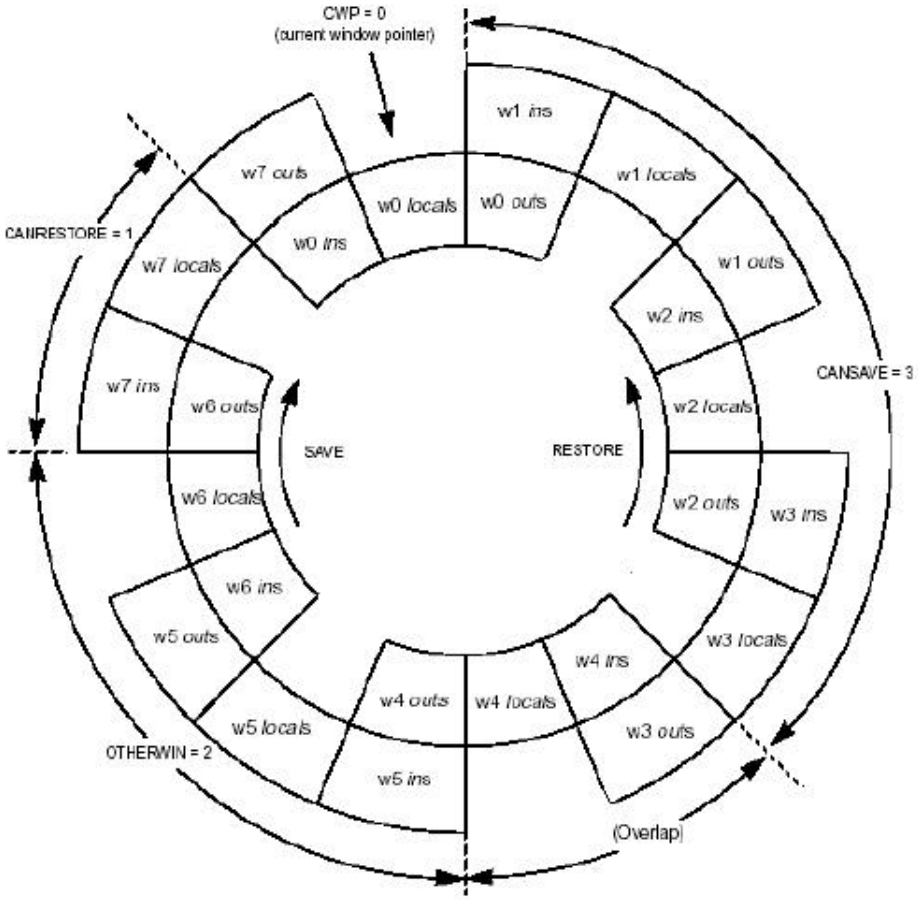


Figura 4 - Registradores Janelados r para Nwindows = 8

As janelas devem ser numeradas continuamente a partir de 0 até NWindows-1. Os endereços dos registradores são dados pela tabela abaixo:

Tabela 2 - Endereçamento da Janela Windowed Register Address *r* Register Address

Windowed Register Address	*r* Register Address
entrada[0] – entrada[7]	*r*[24] – *r*[31]
local[0] – *local*[7]	*r*[16] – *r*[23]
saída[0] – saída[7]	*r*[8] – *r*[15]
global[0] – *global*[7]	*r*[0] – *r*[7]

A janela corrente é dada pelo registrador CWP (*current window pointer*). O CWP é decrementado pela instrução RESTORE e incrementado pela

instrução SAVE. O overflow de janela é detectado pelo registrador CANSAVE e o underflow é detectado pelo registrador CANRESTORE.

b) registradores de status ou controle

- Program counter register (PC)
- Next program counter register (nPC)
- Processor state register (PSTATE)
- Trap base address register (TBA)
- Processor interrupt level register (PIL)
- Current window pointer register (CWP)
- Condition Codes register (CCR)
- Address space identifier register (ASI)
- Hardware clock-tick counter register (TICK)
- Restorable windows register (CANRESTORE)
- Outros registradores de janela (OTHERWIN)
- Registradores para limpeza de janela (CLEANWIN)
- Registrador de status de janela (WSTATE)
- Registrador de Versão (VER)
- Registrador de estado ponto-flutuante (FSR)
- Registrador de registrador de estado ponto-flutuante (FPRS)

3. Histórico dos Sistemas SPARC

A arquitetura RISC, desenvolvida pelo trabalho pioneiro de Dave Patterson na UC Berkeley, veio a transformar-se na arquitetura SPARC que cresceria dentro da indústria até chegar aos microprocessadores de 64 bits. Em 1984, um time de engenheiros da Sun Microsystems incluindo Bill Joy, se reuniu para definir a arquitetura SPARC, baseada profundamente nas especificações RISC de Patterson. A tabela mostra a linha do tempo da SPARC.

Linha do Tempo da arquitetura SPARC - Data / Evento

1984 - David Patterson da UC Berkeley e Bill Joy d Sun Microsystems começam a desenvolver a arquitetura SPARC.

1986 - Sun/Fujitsu implementa o primeiro processador SPARC® SPARC Versão 7™ é publicada.

1987 - Sun-4/260, primeria *workstation* baseada na arquitetura SPARC.

1989 - Nascimento da SPARC *International.*

1990 - SPARC Versão 8 é publicada

1991 - SPARCserver® 600MP, primeiro sistema SPARC-based multiprocessado SPARC LT®, primeiro SPARC-based laptop.

1992 - SuperSPARC® I, primeiro processador SPARC superescalar
Processador SPARClite®
Laptop SPARCbook®
SPARCard® upgrade para PCs

1993 - Processador HyperSPARC® SPARC Versão 9™ é publicada

1994 - Processador SuperSPARC® II
Padrão IEEE 1754-1994 é publicado
ft SPARC® computador tolerante a falhas.

1995 - Processador UltraSPARC® I
Processador SPARC64®
Processador SPARClet®

1996 - Processador TurboSPARC®
1997 - Processador UltraSPARC® II

2000 - Servidor SPARC STAR®

2001 - Processador UltraSPARC® III
Processador SPARC64® IV

2002 - SPARCblade™ computador em uma placa de alta confiabilidade para aplicações de telecomunicações
Laptop GENIALstation

Processador SPARC64® V

Em 1987, quando a Sun introduziu o primeiro computador baseado na arquitetura SPARC, estávamos longe da SPARC que se iria tornar líder em plataformas de hardware de aplicações servidoras. Isto se deu no tempo em que a maioria dos fabricantes de computador estavam tentando introduzir computadores com a tecnologia RISC. As poucas arquiteturas com estas características, derivadas do Unix, que sobreviveram foram: - SPARC, HP's Precision Architecture e Mips da Silicon Graphics. Algumas outras continuaram como aceleradores

gráficos ou de ponto flutuante, como processadores RISC da Intel 860 e 960. O PCRT da IBM foi abandonado e substituído, muitos anos depois, pela família PowerPC, que obteve muito mais sucesso. A família DEC's Alpha também começou muito tempo mais tarde.

Em 1987 a Sun já tinha alcançado uma posição de liderança na venda de workstations com suas duas linhas de produtos baseadas na tecnologia de processadores Motorola 68020, e Intel 386. O 386i posicionou-se como uma estação de trabalho mais barata e de menor desempenho comparado com a família Motorola Sun-3.De fato, dentro de poucos anos, Sun abandonou as plataformas Intel e Motorola e migrou toda sua base de vendas para SPARC.

A Sun e outras empresas mudaram para arquitetura baseada em RISC porque ao criarem sua própria arquitetura sobre RISC, havia o atratativo para que as "OEM" de minicomputadores se livrariam da dependência das duas grandes fabricantes de chips de microprocessador (Intel e Motorola). Vendedores esperavam diferenciar seus produtos. Em 1986, 90% dos PCs usavam chips Intel. O relativo baixo custo de acesso a semicondutor motivou o surgimento da SPARC. Fabricantes de chips precisavam de altos volumes de produção para recuperar o capital investido de centenas de milhões de dólares vislumbraram um caminho para produzir chips chamados "gate arrays", que eram idênticos, exceto pelos últimos passos no processo de produção, que definia ar interconexões. Isto permitia as companias de sistemas de computação produzir chips por alguns milhares de dólares, ao invés dos milhões que eram necessários anteriormente. O renascer do processo de design de chips por sofisticadas ferramentas "computer aided design" (CAD) permitiu que engenheiros projetassem, testassem e simulassem a maioria dos estágios do processador em suas estações de trabalho, sem necessitar conhecimento específico do processo de fabricação. O software permitiu que projetistas trabalhassem no nível de blocos funcionais, sem ter de se preocupar exaustivamente com as interconexões, datapaths e variações do processo. Esta revolução no projeto lógico de hardware foi similar a invenção das linguagens de programação de alto nível que permitiram programadores escrever suas aplicações em termos do problema, podendo se manter afastado de como exatamente o computador iria construir e ligar todo o programa.

No mercado de computação, tinha aumentado a preocupação dos altos custos da obsolescência das aplicações, bem como de hardware e software. Aprendendo com as rápidas mudanças tecnológicas no tempo do início do PC Intel, usuários estavam começando a apreciar padrões de sistema aberto que recebiam suporte de muitos fabricantes o que

ajudava na aumentar o desempenho. Na área de aplicações multi-usuário, apenas um sistema operacional - Unix - parecia ser uma escolha segura. As alternativas proprietárias dos líderes de mercado DEC e IBM eram muito capazes, mas tinham um custo mais alto. Para vendedores, a simplicidade de licença e portabilidade do Unix dentro de seu hardware dava a muitas pequenas companhias a capacidade de oferecer alternativas competitivas sérias.

Tabela 3 – Produtos SPARC com marca registrada

t SPARC®	hyperSPARC®	microSPARC®
MicroSPARC®	SPARC®	SPARCbook®
Micro-SPARC®	SPARC64®	SPARC Compliant®
SoundSPARC®	SPARC64/OS®	SPARC (Stylized)®
SPARC Compliant SCD and Design®	SPARC LT®	SPARCompiler®
SPARCengine®	SPARC O/S®	SPARCplug®
SPARC International®	SPARCluster®	SPARCprinter®
SPARClite®	SPARCluster 1®	SPARCcenter®
SPARClet®	SPARCstorage®	TurboSPARC®
SPARCphone®	SPARCtop®	SuperSPARC®
SPARCserver®	SPARCworks®	SPARCsummit®
SPARCstation®	SPARK®	UltraSPARC Driven & Design®
UltraSPARC Driven & Design (Logo)®	UniSPARC®	UltraSPARC®

4. Evolução da Família SPARC

4.1 Sun 4/260 (Primeira máquina SPARC)

CPUs	1
Clock speed	16.67 MHz
Processor	SF9010
Co-processor	Weitek 1164/1165
MMU	Sun-4, 16 hardware contexts
Processor	501-1274, 501-1491, 501-1522
On-chip cache	
External cache	
Memory	128 MB, 1 GB / process virtual Uses 60ns ECC memory.
Speed	10 MIPS, 1.6 MFLOPS
Chassis type	Deskside
Bus	(12) VME slots
Ports	
Internal storage	
Graphics	
Power supply	
Architecture	sun4
OS's supported	
Notes	First SPARC machine. Code-named "Sunrise". Cache much like Sun-3/2xx, uses same memory boards.

4.2 Unidade Central de Processamento SuperSPARC II

SuperSPARC II usa um pipeline de 4 ciclos, mostrado na tabela5.

Tabela 5 – SuperSPARC II Pipeline Stages

Clock Edge	1	0	1	0	1	0	1	0
SuperSPARC II Pipe Stage	F		D		E		W	

F (Fetch)
Busca instrução na memória e a deposita na fila de instruções buscadas. Maioria das operações de *fetch* ocorrem em um ciclo.

D (Decode, Grouping, Read Rfile)
Seleciona uma instrução candidata a primeira na fila de intruções buscadas. Seleciona os índices dos registradores utilizados, estende palavras para acessos a memórias e branches, checa dependências e insere bolhas quando dependencias não podem ser tratadas com *forwarding*. Gera o endereço de *branch*. Gerencia o próximo PC e PPC (prefetched PC). Soma registrador de endereço para formar endereço virtual.

E (Execute)
Faz execução da IU (Integer Unit), acessa dados do cache, executa comandos de ponto flutuante. Resolve excessões acumuladas no pipeline.

WB (Write Back)
Escreve no *register file* e salva o cache de dados. Durante exceções, salva PC e nPC da instrução que causou erro em %r17 e %r18.

4.3 Unidade Central de Processamento UltraSPARC II

4.3.1 Diagrama de Blocos

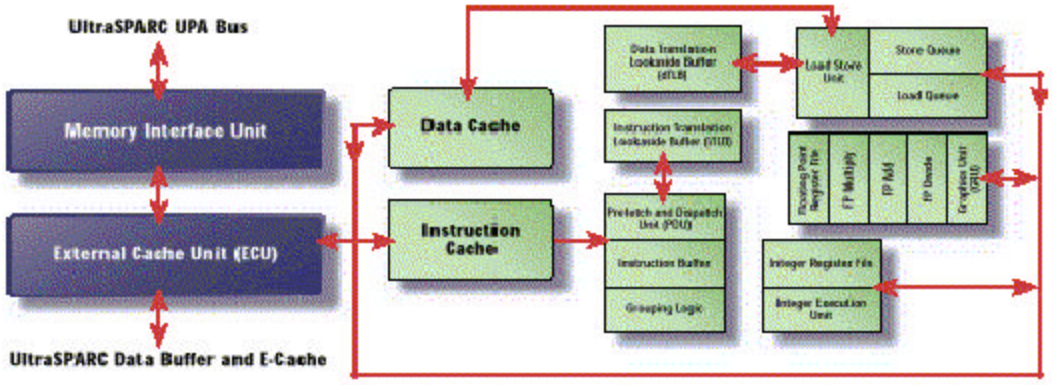


Figura 5 – Diagrama de Blocos de uma CPU UltraSPARC II

4.3.2 Especificação da UltraSPARC II

- Prefetch, branch prediction e dispatch unit (PDU)
- 16-Kbyte instruction cache (I-Cache)
- Memory management unit (MMU) contendo 2 buffers de 64 entradas – TLB de instrução e TLB de dados
- Unidade de execução inteira (IEU) com duas ULAs
- Unidades de Load e Store com gerador de endereçamento separados. Duas unidades.
- Buffer de Load e Store de 16K
- Unidade de Ponto Flutuante com subrotinas independentes para add, divide, multiply e elevar ao quadrado. 2 unidades
- Unidade gráfica conposta por dois pipelines de execução
- Unidade de controle de cache externa
- Unidade de interface de memória, responsável por I/O com a memória principal.
- Implementação de 64bits da arquitetura SPARC V9
- 5.4 milhões de transistores
- Cache on-chip para instruções: 16KB
- Cache on-chip para dados: 16KB
- Cache de nível 2: até 8MB com pico de transmissão de 1.92GB/s
- Clock 360MHz a 480MHz
- Processador de 5 camadas

4.4 Unidade Central de Processamento UltraSPARC III

4.4.1 Diagrama de Blocos

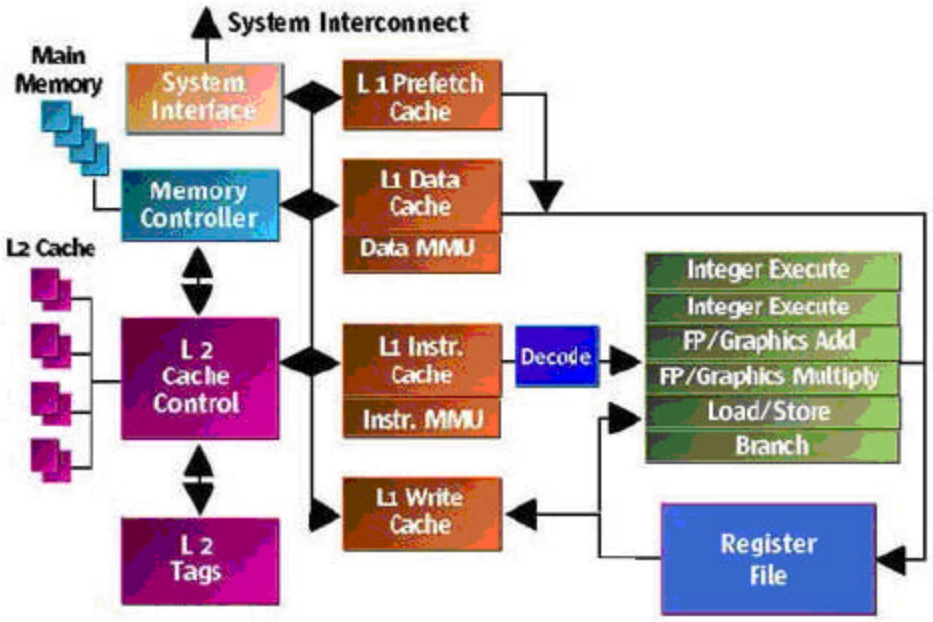


Figura 6 – Diagrama de Blocos de uma CPU UltraSPARC III Cu

4.4.2 Aplicações chave

- Missão crítica
- Computadores Pessoais
- Aplicações de computação intensiva

4.4.3 Especificação da UltraSPARC III

Arquitetura: 64-bit SPARC V9 com conjunto de instruções VIS com poucos tipos de formatos de instrução. Todas as instruções são 32 bits, e são alinhadas em vizinhanças de 32 bits na memória;

Somente as instruções load e store têm a acesso a memória e realizam IO (arquitetura do tipo *load/store)*

Poucos modos de endereçamento: o endereço na memória é dado como "registrador + registrador" ou "registrador + imediato";

Possui operações com três registradores, sendo que dois deles são operandos e o terceiro onde é colocado o resultado;

Grande arquivo de registradores "janelados". Podem ser usados como um cache para argumentos de um procedimento, valores locais e como retorno de endereço. Em qualquer instante o programa vê 8 registradores globais de inteiro mais uma janela de 24 registradores do arquivo de registradores.

A arquitetura SPARC-V9 provê compatibilidade com o conjunto de instruções IEEE 754 para ponto-flutuante, operando em um arquivo de registradores separado. Este arquivo provê: 32 registradores para precisão simples (32 bits); 32 registradores para precisão dupla (64 bits); 16 registradores para precisão quádrupla (128 bits) ou mista.

Predição de desvio (branch) dando ao hardware uma grande probabilidade de manter o pipeline do processador cheio.

4-way Superscalar: (sustenta 4 instruções por / ciclo), provê paralelismo para uma máquina superscalar, tratando várias instruções ao mesmo tempo.

Pipeline sem Stall de 14 estágios o 6pipelines de execução (2 integer, 2 FP/VIS[tm] Instruction Set, 1 load/store, 1 branch)

Instruções de multiplicação e divisão de inteiros feita em 64-bits.

Contém 16 registradores adicionais ponto-flutuante, totalizando 32 registradores. Esse aumento reduz o tráfego em memória, permitindo programas rodarem mais rápido. Esses 16 registradores são endereçáveis com 8 registradores de precisão 4 (eigth quad-precision registers)

Suporta o formato 128-bits para o formato ponto-flutuante

Suporta quatro registradores de condição ponto-flutuante, onde o SPARC-V8 suportava apenas um, o que gerava dependência e espera para que uma única condição seja atualizada.

Instruções load e store de ponto-flutuante feita com 4 palavras (quad-word)

Controlador de Memória Integrado capaz de acessar 16GB de memória a 2.4 GB/s

Cache de nível 1: Instruções (32KB) & dados (64KB)

Cache de nível 2: Até 8MB externo

Aspectos RAS avançados (Reliability, Availability and Serviceability)

Escalabilidade : Mais de 1000 CPUs desenvolvida para três aspectos: tempo, implementações e o número de processadores

Barramento: Sun Fireplane interconectado a 150MHz

Frequência do clock: 900MHz a 1.2GHz

Memória principal acessável por processador: 16GB

Sistema Operacional: Solaris 8 ou superior

Execução especulativa de branchs e loads

Execução a clocks mais baixos (de 1/2 a 1/32) para economizar energia

Arbitragem do barramento de forma distribuída

Dissipação de energia máxima 65 watts

Tensão de 1.6 volts

Processo de 0.15 micron e 7 camadas

4.4.4 Pico da Taxa de Instruções

O processador sustenta 4 instruções a cada ciclo de clock que é o máximo número de instruções que podem ser buscadas (*fetched*) do cache em um ciclo de clock. Um total de 16 instruções podem ser enfileiradas, esperando que uma unidade de execução apropriada se torne disponível. Existem 6 unidades de execução paralelas: 2 ULAs inteiras (idênticas) , 1 unidade de branch, 1 unidade de load/store e 2 unidades de ponto flutuante (uma para soma e subtração e uma para multiplicação e divisão)

4.4.5 Pipeline para altas taxas de clock

Para alcançar taxas de clock mais altas, o pipeline de execução do processador UltraSPARC III é segmentado em 14 estágios separados. Como uma consequência, este pipeline é capaz de trabalhar desde sua frequencia inicial de 600 MHz até a freqüência corrente de 1050 MHz e poderá passar de 1500MHz nas próximas gerações de tecnologia dos semicondutores.

4.4.6 Gerenciamento de Branch

O processador UltraSPARC III implementa um avançado mecanismo de *branch prediction* baseado em um histórico de 4KBs que precisamente prediz se um *branch* será tomado ou não com acerto de cerca de 95%. Erro na predição de um branch tem uma penalidade de 7 ciclos. Entretanto, UltraSPARC III toma alguns passos para diminuir o impacto de *missprediction* para em torno de 4.5 ciclos.

4.4.7 Organização da Memória

Como outros processadores 64bits, o processador UltraSPARC III deixa os programas acessarem um enorme espaço de endereçamento virtual, 16 quintilhões de bytes, 4 bilhões de vezes maior do que os 4GBs limitados por endereçamento de 32 bits. Para dispositivos fora do chip, ele produz endereçamentos de 43 bits – suficiente para acessar 8 TeraBytes de memória física. O processador UltraSPARC III admite dois níveis de cache. O primeiro nível [L1] consiste de quatro caches separadas, duas grandes e duas pequenas. O segundo nível [L2] consiste de apenas um chip grande.

As duas caches grandes do nível L1 cuidam de intrução e dados, respectivamente, as duas caches pequenas do nível L1 são caches de *prefetch*, utilizadas principalmente para execução especulativa de intruções do tipo load e outra para write. Estas caches diminuiem significativamente a largura de banda requerida pelo sistema. Todas as caches L1 estão no chip.

A cache de nível L2 é uma cache unificada para intruções e dados. É feita sobre a tecnologia SRAM em um chip a parte, mas as tags são mantidas dentro do processador, para acelerar o acesso.

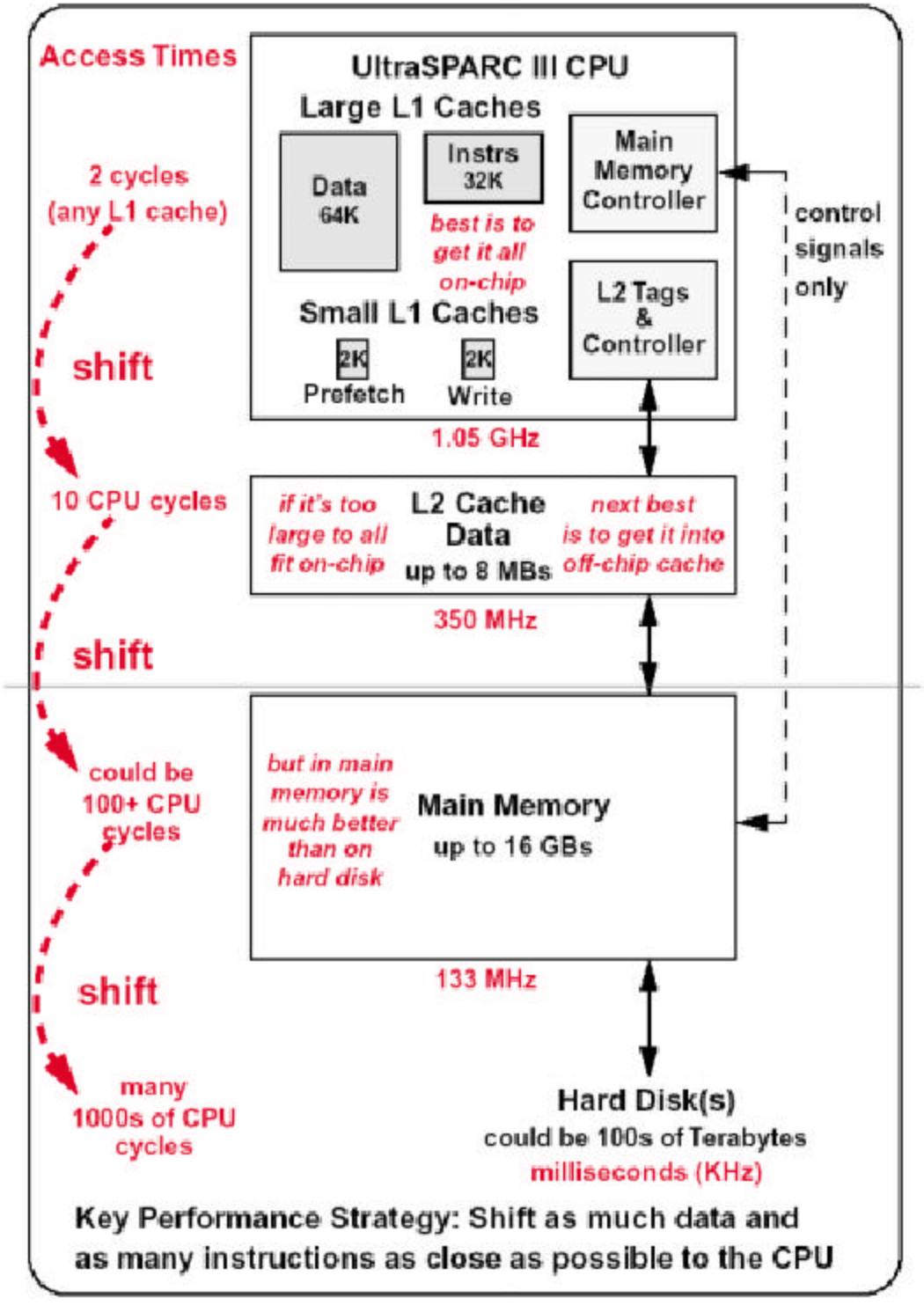
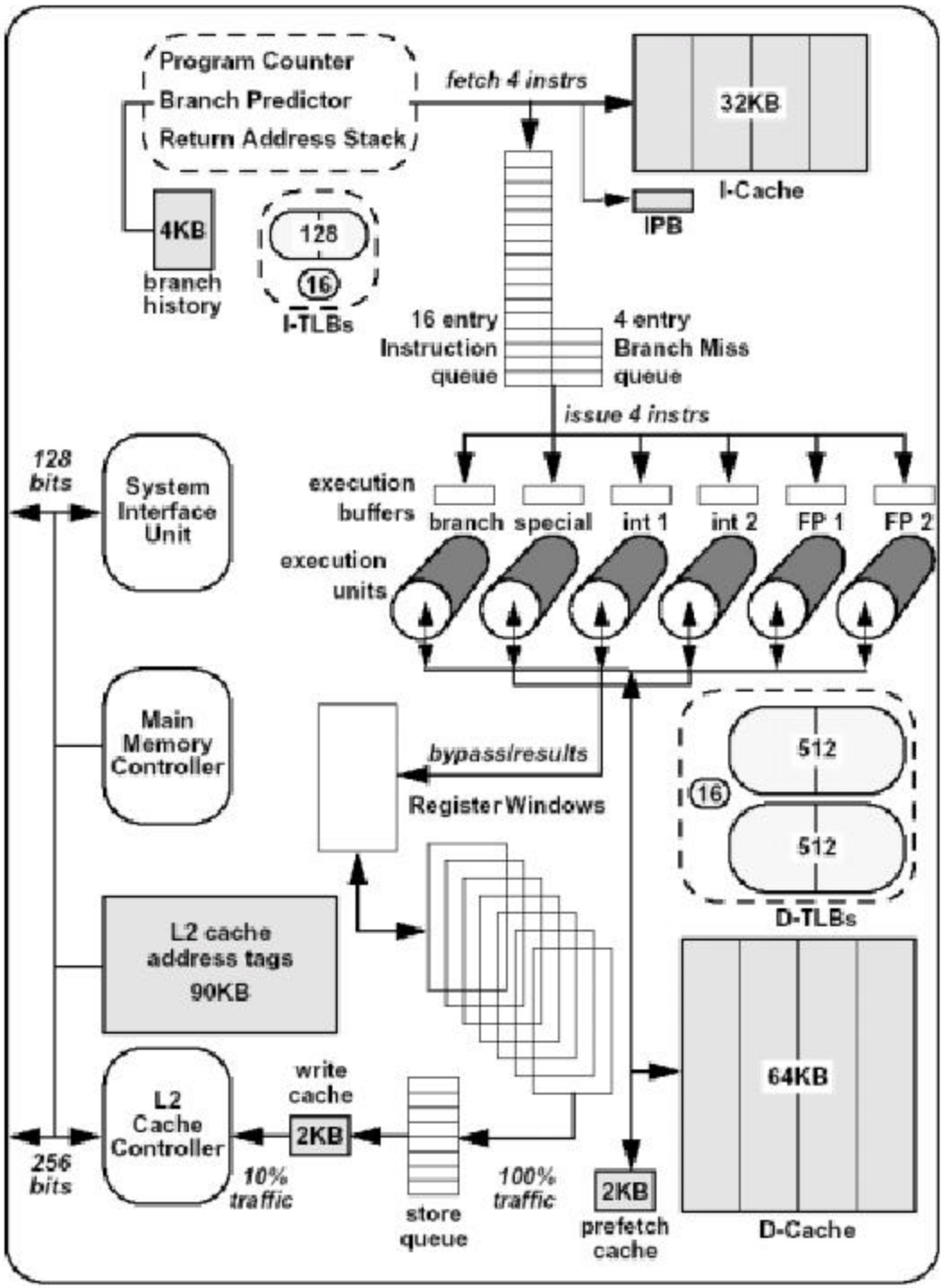


Figura 7 - Hierarquia de memória do Processador UltraSPARC III Cu

4.4.8 Datapath do UltraSPARC III

Figura 8 - Datapath do UltraSPARC III Cu



4.4.9 Unidade de Inteiros (IU)

A Unidade de Inteiros (*Integer Unit*) executa instruções aritméticas de números inteiros e computa os endereços de memória para as instruções it load e it store. Essa unidade também mantém os contadores de programa (*program counters*) e controla a execução das instruções para a unidade FPU.

Uma implementação da IU pode conter de 64 a 528 registradores *r* de 64-bit. Eles são particionados em 8 registradores globais, 8 registradores alternativos, mais um número de registradores dependentes.

Esses registradores dependentes são uma pilha circular com 3 a 32 conjuntos de 16 registradores cada, conhecida como janelas de registradores. Por exemplo, uma janela de 24- registradores consiste de um conjunto de 16 registradores divididos em 8 registradores inteiros e 8 globais, e mais 8 registradores externos adjacentes aos 8 internos.

A janela corrente é especificada pelo registrador CWP (*current window pointer*). O processador detecta o transbordamento de janela via o registrador CANSAVE e executa as exceções via o registrador CANRESTORE.

Sempre que a IU acessa uma instrução ou dado na memória, ele anexa um identificador ASI (*address space identifier*) ao endereço. Todos os acessos à instrução e quase todos os acessos aos dados anexam um ASI implícito, mas algumas instruções permitem a inclusão de um ASI explícito com um campo imediato na instrução ou a partir do registrador ASI.

O ASI determina a ordem do byte de acesso. Na arquitetura SPARC todas as instruções são acessadas em ordem de byte “big-endian” e os dados podem ser referenciados como ordem “big ou little-endian”.

4.4.10 Unidade de Ponto-flutuante (FPU)

A Unidade de Ponto Flutuante (FPU) possui 32 registradores 32-bits de precisão simples, 32 registradores 64-bits de precisão dupla e 16 registradores 128-bits de precisão quádrupla.

Os valores de precisão dupla ocupam pares de registradores de precisão simples, e os valores de precisão quádrupla ocupam um grupo de quatro

registradores de precisão simples. Os 32 registradores de precisão simples, a metade inferior dos registradores de precisão dupla e a metade dos registradores de precisão quádrupla cobrem uma às outras. A metade superior dos registradores de precisão dupla e quádrupla cobrem umas às outras, mas não o fazem com os de precisão simples.

As instruções *loadstore* de ponto-flutuante são usadas para mover dados entre FPU e a memória. O endereço de memória é calculado por IU.

A arquitetura SPARC utiliza o padrão IEEE Std 754-1985 para 32 e 64 bits e o padrão IEEE Std 1596.5-1992 para 128 bits. Se uma FPU não está presente ou não está habilitada uma tentativa de executar uma instrução ponto-flutuante gera uma trap - ação, fp_disabled em resposta a presença dessa exceção.

Neste caso o software deve ou habilitar a FPU e re-executar a instrução de *trap* ou emular a instrução de *trap*.

4.4.11 Melhorias feitas no SPARC-V8

a) Endereços virtuais 64-bit e dados inteiros de 64-bits

b) Todos os registradores inteiros no SPARC-V8 foram estendidos de 32 bits para 64 bits. A compatibilidade com SPARC-V8 é feita de forma a possibilitar que programas compilados para 32-bit em um processador SPARC-V8 seja executado em um SPARC-V9 utilizando o princípio do complemento de 2.

c) As operações em 32-bits continuarão a gerar os mesmos valores, ou seja, 32-bits. Em alguns casos são adicionadas novas instruções para operar valores 64-bits. Por exemplo, instruções *shift* que agora tem a forma adicional 64-bits.

d) Endereços virtuais 64-bit e dados inteiros de 64-bits

e) Todos os registradores inteiros no SPARC-V8 foram estendidos de 32 bits para 64 bits. A compatibilidade com SPARC-V8 é feita de forma a possibilitar que programas compilados para 32-bit em um processador SPARC-V8 seja executado em um SPARC-V9 utilizando o princípio do complemento de 2.

f) As operações em 32-bits continuarão a gerar os mesmos valores, ou seja, 32-bits. Em alguns casos são adicionadas novas instruções para operar valores 64-bits. Por exemplo, instruções *shift* que agora tem a forma adicional 64-bits.

g) Performance melhorada do sistema.

5. Análise de Desempenho (SPEC)

A SPEC (Standard Performance Evaluation Corporation) é uma empresa sem fins lucrativos; registrada na Califórnia; formada para estabelecer, manter e endossar um conjunto de *benchmarks* relevantes e padronizados, que podem ser aplicados na mais nova geração de computadores de alta performance. Seus fundadores acreditam que a comunidade de usuários será beneficiada com os testes gerados que servirão como referência durante o processo de avaliação da apliação.

O SPEC possui duas funções:

a) desenvolver suítes de *benchmarks* para medição da perfomance de um computador. Essas suítes são pacotes com código fonte e ferramentas extensivamente testadas para garantir portabilidade. As suítes estão disponíveis para o publico, que paga apenas taxas administrativas.

b) Os resultados são publicados no *The SPEC Newsletter* and *The GPC Quarterly* e disponíveis eletronicamente em [9].

A SPEC fornece avaliações para diversos fabricantes de processadores que utilizam a arquitetura SPARC. As tabelas 6 e 7 mostram uma avaliação realizada com processadores cuja arquitetura é SPARC. A tabela 8 mostra uma avaliação realizada com processadores cuja arquitetura é RISC.

Tabela 6 – SPECint 2000 de processadores com Arquitetura SPARC

Fabricante	Processador	Taxa Clock (MHz)	SPECint
Sun	Sun Blade 150 (UltraSPARC IIi)	550	217
	Sun Blade 1000 Modelo 1600 (UltraSPARC III)	600	313
	Sun Blade 1000 Modelo 1750 (UltraSPARC III)	750	396
Fujitsu	PrimePower 600	500	354
	PrimePower 200	600	406
	PrimePower 400	700	521

Figura 9 - SPECint de processadores com arquitetura SPARC

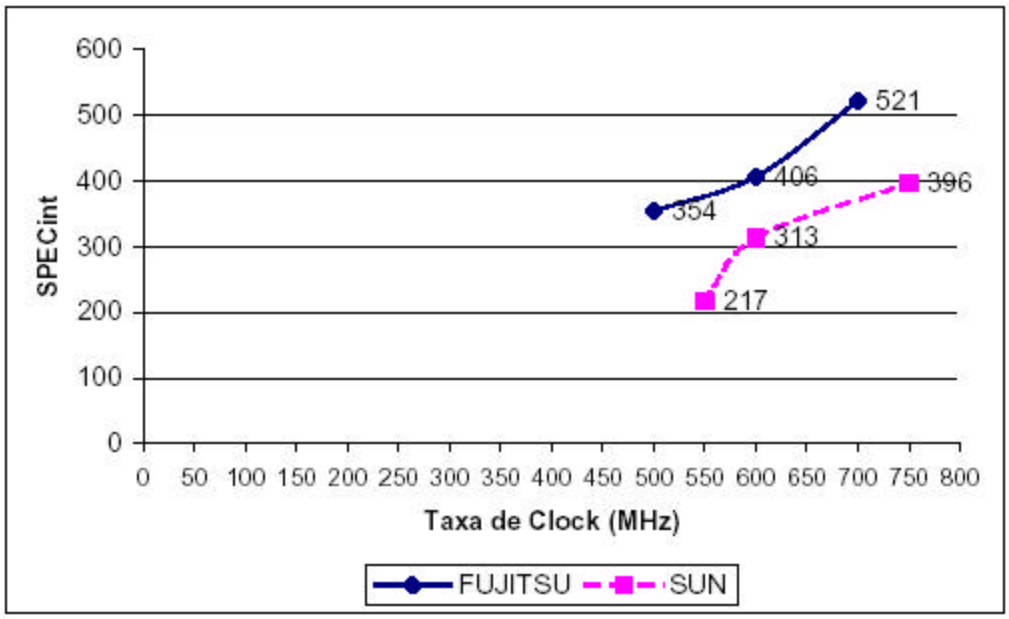


Tabela 7 – SPECfp 2000 de processadores com Arquitetura SPARC

Fabricante	Processador	Taxa Clock (MHz)	SPECint
Sun	Sun Blade 150 (UltraSPARC IIi)	550	254
	Sun Blade 1000 Modelo 1600 (UltraSPARC III)	600	260
	Sun Blade 1000 Modelo 1750 (UltraSPARC III)	750	395
Fujitsu	PrimePower 600	500	-
	PrimePower 200	600	403
	PrimePower 400	700	506

Outra comparação pode ser realizada utilizando-se outros processadores oriundos também da arquitetura RISC, como por exemplo a arquitetura PowerPC e arquitetura SPARC. Porém são encontradas dificuldades em compararmos efetivamente os processadores uma vez que os mesmos não possuem todas as taxas de clock selecionadas.

Tabela 8
SPECint 2000 de processadores oriundos da Arquitetura RISC

Fabricante	Processador	Taxa Clock (MHz)	SPECint
Sun	Sun Blade 150 (UltraSPARC IIi)	550	254
	Sun Blade 1000 Modelo 1600 (UltraSPARC III)	600	260
Fujitsu	PrimePower 600	500	-
	PrimePower 200	600	403
IBM	IBM PowerPC RS64-III RS/6000 Modelo 7026-F80	450	234
	IBM PowerPC RS64-III RS/6000 Modelo 7026-M80	500	275

5.1 Informações de Mercado

Com a introdução da workstation Sun Blade 2000 a Sun continua mantendo sua liderança no mercado de workstations 64-bits, a partir da EDA (eletronic design automation) e MCAE/MCAD (mechanical

computer-aided engineering/design). Essa nova workstation possui dois disk-drivers de 73GB interno FC-AL e 4.8GB por segundo de throughput, e foi desenvolvida para computar aplicações que requerem alta performance como as gráficas.

Contendo o processador UltraSPARC III Cu 900 MHz, essa workstation é mais rápida do que todos os concorrentes 64-bits e 70% mais rápida do que a Intellistation POWER 265 450 MHz da IMB.

6. Introdução ao Sistema Operacional Solaris

Solaris é o nome da Sun para o seu ambiente UNIX, incluindo o sistema operacional UNIX, sistemas de janelas, entre outras coisas. Não é o sistema operacional, mas o ambiente operacional. Os componentes do sistema operacional do Solaris são SunOS 4.x e SunOS 5.x para Solaris 1.x e 2.x respectivamente.

Por 10 anos, o desenvolvimento do UNIX foi essencialmente confinado no Bell Labs. As versões iniciais eram chamadas de "Version n" ou "Nth Edition" (dos manuais).

Além dessa implementação, existe uma segunda implementada pela University of Berkeley conhecida como BSD UNIX.

Algumas datas:

> 1991 - SunSoft lança Solaris 2 e introduz Solaris para a plataforma Intel x86.

> Março/95 - Sun anuncia Solaris DT para a Enterprise.

> Junho/95 - Sun introduz o Solaris 2.5. Sun junta-se com a IBM para portabilizar o Solaris para Powerpc.

> Agosto/95 - Sun e Informix criam nova tecnologia de bando de dados no solaris X86.

> Setembro/95 - Sun anuncia novo servidor Wabi, Solaris Base Server.

6.1 Versões e suas atualizações

Solaris 2 - lançado pela SunSoft, é a primeira versão otimizada do SunOS, baseada no UNIX SRV4.

Solaris 2.2 - As melhorias no OpenWindows incluem Image Tool, o Properties Tool incrementado, capacidades internacionais integradas e muitos incrementos fáceis de usar.

Solaris 2.3 - Uma implementação assíncrona do PPP (Point-to-Point Protocol), incluído no protocolo da Internet, que habilita enlace de comunicação usando modems e linhas de telefone. Um mecanismo de cache não volátil para aumentar a performance de sistemas de arquivo usando um disco pequeno, rápido e local.

Solaris 2.4 - Adicionado ao sistema operacional estão quatro linguagens européias e quatro asiáticas, bem como o espanhol latino-americano e o inglês americano, para desktop e configurações de produtos workgroup server. Solaris 2.4 roda em todas as máquinas SPARC.

Solaris 2.5 - Esta versão inclui: SunOS 5.5, OW 3.5 e DeskSet 3.5.

SunOS 5 é baseado na versão System V do UNIX. Solaris 2.5 roda em todas as máquinas SPARC exceto as da série Sun4, ou seja, Sun4/110, Sun 4/280,...

Solaris 2.6 - Esta versão inclui: SunOS 5.6

Solaris 7 (2.7) - Esta versão inclui: SunOS 5.7

Solaris 8 – O Ambiente Operacional Solaris 8 é o sistema líder para servidores UNIX, sendo o de maior prestígio. Teve mundialmente mais das 30% de entregas desse setor durante o ano 2000, e está na dianteira com mais do dobro da fatia de mercado do seu concorrente.

Solaris 9 – Esta versão ajuda a acelerar a implementação de serviços de Web com sua pilha central de softwares integrados.

6.2 Overview do Ambiente Operacional

O Sistema Operacional é um set de programas que gerencia todas as operações do computador e providencia uma interface entre o usuário e os recursos do sistema.

As aplicações Common Desktop Enviroment (CDE) e OpenWindows. O CDE é o ambiente mais agradável ao usuário, com interface gráfica mais rica, possui quatro desktops, além de opções de cores, backgrounds e uma barra localizada no campo inferior da tela, que disponibiliza os principais itens de aplicativos.

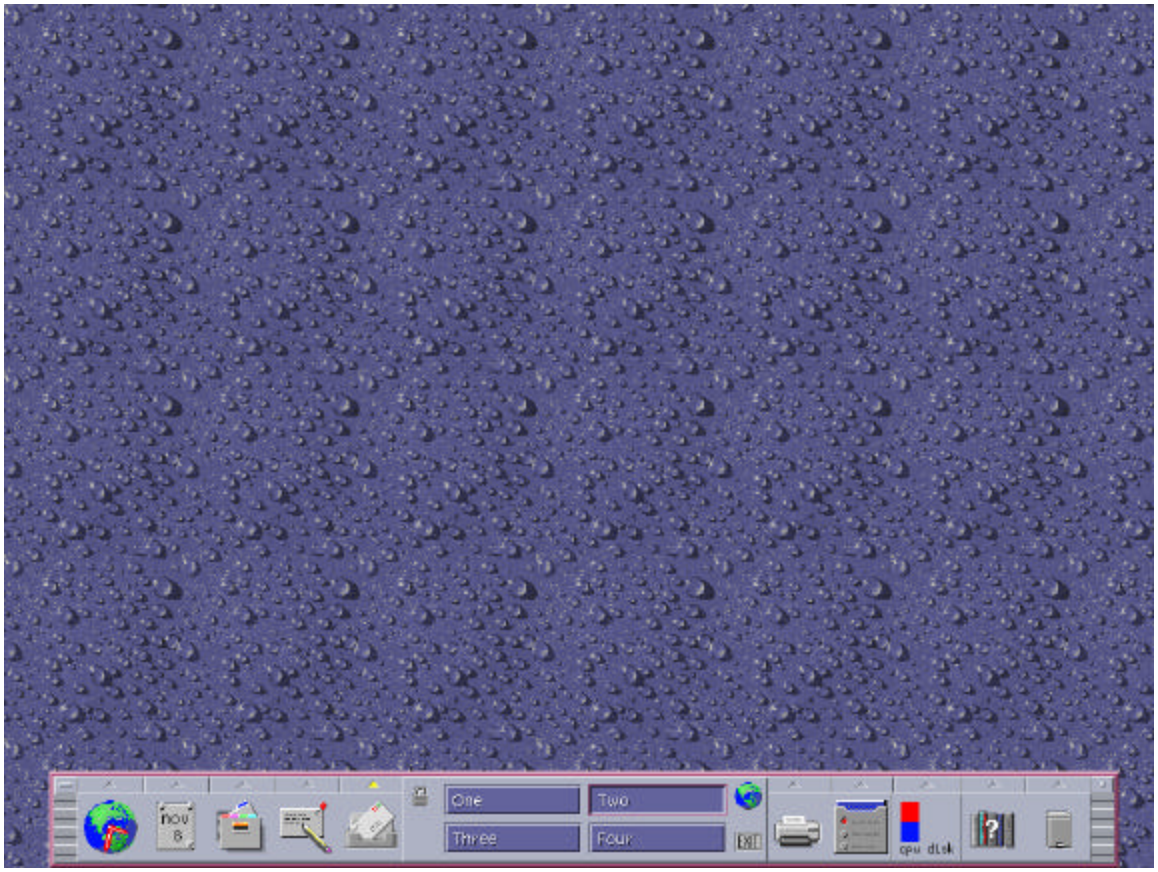


Figura 10 – Janela do CDE

O OpenWindows atende a usuários que não tenham muita necessidade da parte gráfica. Ele utiliza basicamente os terminais e seu menu suspenso para executar seus aplicativos. Vide figura 11.

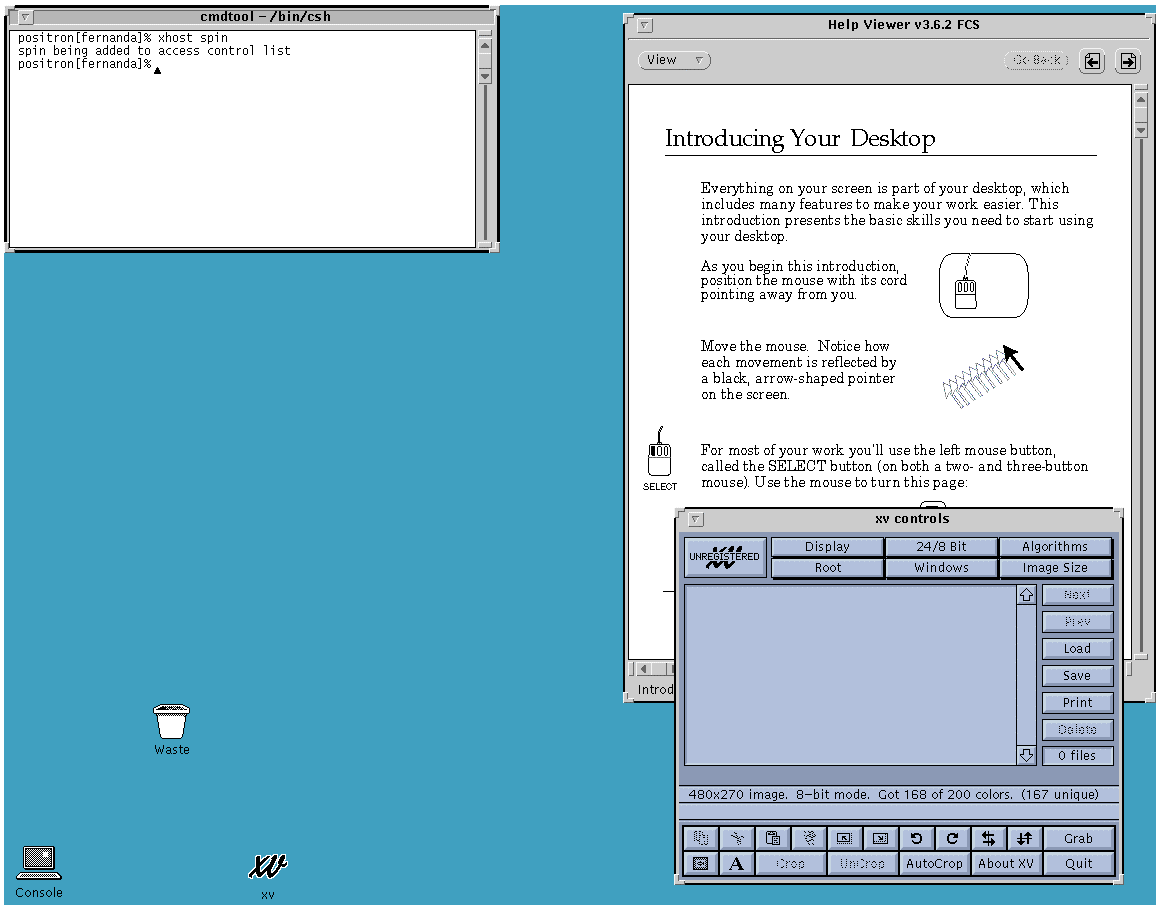


Figura 11 – Ambiente OpenWindows

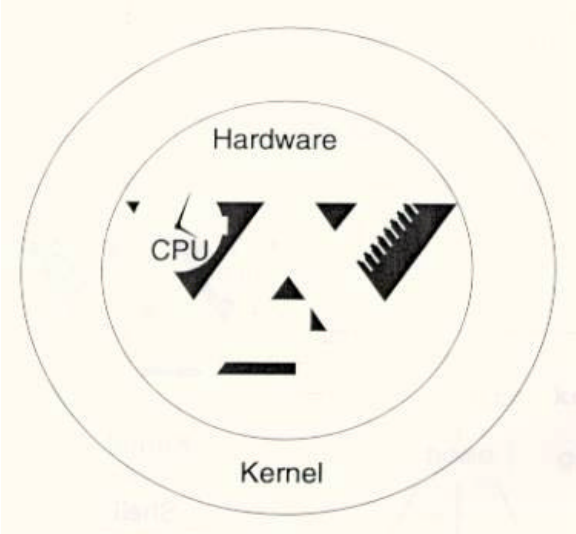
7. Partes Principais do Sistema Operacional:

7.1 Kernel

O Kernel é o núcleo do sistema operacional.
O Kernel desempenha as seguintes funções:

Gerencia dispositivos, memória, processos e daemons.

Controle de funções (transmissão de informações) entre os programas (utilitários) e o hardware.



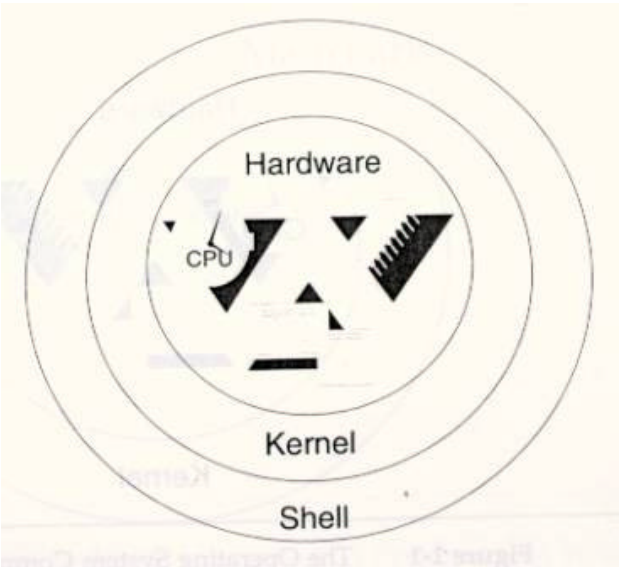
Agenda e executa todos os comandos.

Gerencia funções tais como:

Swap Space – Uma parte reservada do disco que o kernel usa durante o processamento.

Daemons – Processos que executam tarefas do sistema.

7.2 Shell



Um Shell é uma interface entre o usuário e o kernel. Ela age como um interpretador ou tradutor de comandos. O Shell interpreta o que você digitou e envia o executável para o kernel.

Os Shell

Existem 3 shells disponíveis no pacote de instalação do Solaris.

Bourne Shell: O Shell padrão do Solaris. Ele foi desenvolvido para o ambiente AT&T UNIX.

C Shell: A sintaxe do C Shell é similar a linguagem de programação C. Similar ao

Korn Shell.

Korn Shell: Adiciona algumas características em relação ao Shell Bourne.

Também existem os chamdos “shells inteligentes´que tem entre suas ferramentas, voltar comandos já anteriormente dados, bastando apenas navegar com as setas do teclado. Ex: tcsh.

8. Gerenciamento de Memória

Cabe ao kernel do Solaris a responsabilidade por diversas funções de gerência de memória, como controlar que partes da memória estão em uso e que partes não estão, alocar memória para processos quando eles necessitarem, liberar quando eles terminarem e gerenciar a troca entre a memória principal e o disco quando a memória principal for muito pequena para armazenar todos os processos.

Em sistemas de tempo compartilhado, o gerenciador de memória é responsável pela suspensão e retomada de um processo em intervalos de tempos distintos. Na espera pela execução, os processos ficam na memória até que seja identificado o processo ativo que, a partir desse momento, terá sua execução retomada. O gerenciador também reconhece os processos que estão em estado de espera por entrada/saída para que estes não sejam incluídos na partilha de tempo de execução.

O escalonamento de memória do Solaris define as prioridades dos processos, cujo cálculo é referente ao tempo de execução acumulado. Os processos que em execução acumularam muito tempo terão prioridade menor do que processos que ainda não foram executados.

No Solaris, além do swapping - técnica que seleciona alguns processos para retirar da memória caso esta esteja cheia - é implementado a

gerência de memória virtual com paginação por demanda. Nesse esquema, páginas do processo são trazidas do disco para a memória somente quando são referenciadas.

Periodicamente o sistema é ativado para verificar se a quantidade de páginas livres é insuficiente. Nesse caso, o sistema inicia o trabalho de liberação de páginas para recompor a lista de páginas livres.

Quando necessário, o Solaris usa o algoritmo de procura circular para a liberação de páginas. Inicialmente todas as páginas estão marcadas como não utilizadas (bit de acesso igual a zero). Sempre que uma página é referenciada, o seu bit de acesso é ligado (igual a um). De tempos em tempos o sistema varre a lista de páginas, verificando o bit de acesso de cada uma. As páginas que não foram utilizadas são liberadas para a lista de páginas livres. No caso de páginas contendo dados, o sistema verifica também se a página foi modificada para, nesse caso, gravá-la em disco.

9. Sistema de Arquivos

O sistema de arquivos do Solaris é baseado em uma estrutura de diretórios em árvore, não existindo dependência entre a estrutura lógica desses diretórios e o local onde os arquivos estão fisicamente armazenados. Esse modelo permite que uma estrutura seja formada por diferentes discos, inclusive em estações remotas.

Utilizando uma arquitetura denominada Virtual File System (VFS), o Solaris proporciona uma interface padrão para diferentes tipos de sistemas de arquivos, uma vez que essa arquitetura permite ao kernel do sistema controlar operações básicas como ler, escrever ou listar arquivos, sem que seja necessário um conhecimento do tipo de sistema de arquivos, tanto pelo usuário quanto pelo programa.

Existem três tipos de arquivos no Solaris: diretórios, que podem conter arquivos ou outros diretórios; arquivos regulares, contendo qualquer tipo de dado que o usuário deseje; e arquivos especiais, que, como já visto, estão associados a dispositivos de entrada/saída (locais ou remotos).

O Solaris suporta três tipos de sistema de arquivos: sistema de arquivos baseados em disco, que podem ser escritos em diferentes formatos e são armazenados fisicamente em discos flexíveis, discos rígidos ou CD-ROMs; sistema de arquivos virtual, baseados em memória para

proporcionar acesso ao núcleo do sistema sem utilizar espaço em disco; e sistema de arquivos baseado em rede, que são acessados através da rede.

Existem dois tipos de sistema de arquivos baseados em rede, o Network File System (NFS) e o Remote File Sharing (RFS). O NFS habilita computadores e arquiteturas diferentes - utilizando diferentes sistemas operacionais - a compartilhar arquivos através de uma rede. Dessa forma, qualquer computador tem acesso aos arquivos de outro computador. A diferença entre o NFS e o RFS, é que, enquanto o primeiro gera um sistema de arquivos genérico, este último provém uma cópia exata de um sistema de arquivos UNIX.

Por ser um sistema operacional multiusuário, o Solaris necessita de segurança para o sistema de arquivos. Cada arquivo apresenta um nível de proteção definido pela categoria do usuário. Todo arquivo ou diretório possui um user que pertence a um grupo. Qualquer usuário que não seja dono do arquivo e não pertença ao respectivo grupo, enquadra-se na categoria others. Por fim, o administrador do sistema, chamado de root, tem acesso irrestrito a todos os arquivos. Dependendo da categoria do usuário, três tipos de acesso podem ser concedidos, read, write ou execute.

10. Entrada e Saída (I/O)

A independência de cada dispositivo de entrada e saída é um dos fatores mais importantes no Solaris. Dessa forma, um processo pode acessar um arquivo em disco tão facilmente quanto um terminal ou uma impressora.

Todos os requerimentos de entrada e saída são manipulados sincronicamente, ou seja, um processo que solicita uma entrada, por exemplo, é suspenso a partir do momento dessa solicitação e liberado quando a entrada tiver sido completada.

A gerência de entrada e saída no Solaris é implementada por drivers, sendo necessário um driver para cada dispositivo. Esses drivers são acoplados ao sistema operacional e, uma vez acrescentado um novo dispositivo, um driver correspondente será acoplado ao kernel.

O Solaris trabalha com dois tipos de drivers de entrada e saída: driver de bloco, onde a transmissão é feita por blocos e normalmente está associada a dispositivos com altas taxas de transferência entre esse

dispositivo e a memória; e driver de terminal, cuja transmissão é feita caracter por caracter e é usado em dispositivos mais lentos.

No caso do driver de bloco, sempre que um processo solicita uma transferência, o kernel verifica se o bloco já está na memória ou não e, em seguida, o sistema transfere o bloco solicitado para o dispositivo de entrada e saída. Blocos freqüentemente utilizados tendem a permanecer na memória, reduzindo, portanto, o tráfego de entrada e saída.

O driver de terminal é utilizado por todos os dispositivos que não se ajustam ao modelo de blocos. Contudo, a maioria dos dispositivos que possuem a interface estruturada para o driver de bloco, também possui a interface de terminal.

O acesso aos dispositivos de entrada e saída é integrado ao sistema de arquivos através de arquivos especiais. Esses arquivos podem ser acessados da mesma forma que qualquer outro arquivo, utilizando primitivas de leitura e gravação.

11. Processos

O Solaris é um sistema multiprogramável, onde cada usuário pode ter vários processos ativos simultaneamente. Em um grande sistema, podem chegar a existir milhares de processos ativos ao mesmo tempo. O kernel do sistema é responsável pelo controle desses processos, fornecendo primitivas que cuidam, por exemplo, da criação e da gerência de processos.

Inicialmente, quando se dá a ativação do sistema, é criado o processo 0, que por sua vez cria o processo 1, conhecido como init. Este é o ancestral de todos os outros processos.

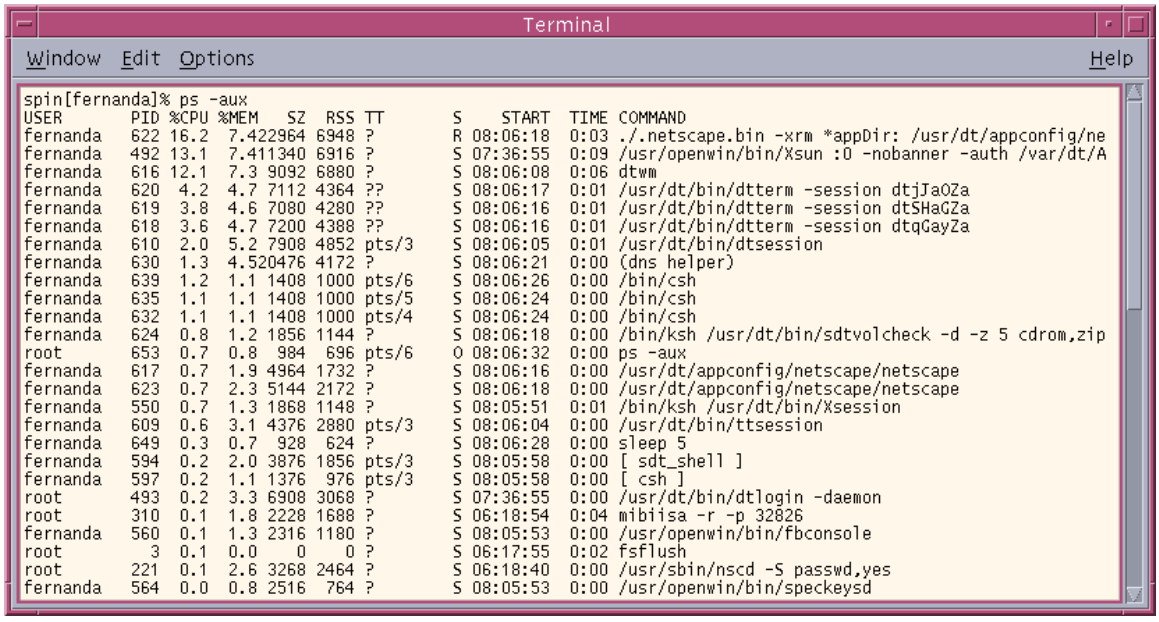
Os processos são criados pela primitiva de sistema fork. Essa função, ao ser chamada por um processo em execução (processo-pai), cria uma cópia igual desse processo (processo-filho) - um processo-pai pode ter vários processos-filhos e estes também podem ter seus processos-filhos. A partir daí, tanto o processo-pai quanto o processo-filho têm seu próprio espaço de endereçamento. Dessa forma, as variáveis de um não são visíveis ao outro e vice-versa.

Os processos são identificados mediante um código de identificação, pid, que é fornecido pelo processo-pai ao processo-filho. A comunicação entre os processos é feita por um mecanismo de troca de mensagens

que utiliza canais de comunicação chamados pipes. Através dos pipes, a saída de um comando é direcionada para a entrada de outro, sem a utilização de arquivos temporários.

Os processos no Solaris possuem duas estruturas-chave: a tabela de processos, que contém informações como número do processo, modo e prioridade, entre outras; e a estrutura de usuário, contendo informações que não são necessárias quando o processo não estiver fisicamente na memória principal.

A gerência de processos do Solaris é usada para mostrar os processos que estão sendo executados numa determinada estação de trabalho, além de parar e reinicializar processos e investigar e depurar processos irregulares (processos que não estão realizando o trabalho esperado).



O gerenciador usa monitores e semáforos como mecanismos de exclusão mútua, e escalonamento circular com múltiplas filas, pra indicar a prioridade de cada processo. Processos sendo executados no modo usuário tem menor prioridade que os processos no modo kernel.

12. Solaris 9

12.1 Solaris 9

O UNIX® Número 1 ficou ainda melhor. O ambiente operacional (AO) Solaris[tm] 9 transforma o sistema operacional em uma plataforma de serviços com a integração de uma Plataforma Java[tm] 2 Enterprise Edition (J2EE[tm]) padrão, servidor de aplicativos e servidor de diretórios Lightweight Directory Access Protocol (LDAP), fornecendo a base para o gerenciamento de identidades. A integração desses e de outros importantes produtos, como os softwares Solaris Volume Manager e Resource Manager do Solaris 9, com o AO Solaris 9 representa uma economia para os clientes em termos de aquisição de software e reduz significativamente o tempo de implementação e gerenciamento de sistemas.

"Com o lançamento do ambiente operacional Solaris 9, a Sun está fornecendo muito mais do que um sistema operacional. Estamos mudando o jogo e oferecendo a melhor plataforma de serviços para o mundo das redes", diz Anil Gadre, vice-presidente e gerente geral do Software Solaris da Sun Microsystems. "O AO Solaris 9 foi construído para padrões abertos de Internet, o que significa melhor proteção de investimento para nossos clientes. Por ser a base da arquitetura Sun[tm] ONE, o ambiente operacional Solaris 9 ajuda a acelerar a implementação de serviços de Web com sua pilha central de softwares integrados".

12.2 Redução do Custo Total das Operações

O ambiente operacional Solaris 9 ajuda a reduzir o custo total das operações por meio de maior integração, gerenciabilidade, segurança, desempenho e disponibilidade.

12.3 Software Integrado

O ambiente operacional Solaris 9 ajuda a reduzir o custo das operações, diminuindo significativamente as despesas de aquisição dos principais aplicativos de software de infra-estrutura. Essa integração com a plataforma Sun Open Net Environment (Sun ONE) proporciona novas eficiências e novas formas de computação. A prestação de serviços por meio da plataforma Sun ONE significa integrar servidores Web, de

aplicação e de diretórios, bem como sistemas de arquivo, gerenciamento de volume, segurança e dimensionamento.

Com o servidor de diretórios Sun ONE totalmente integrado e a inclusão do servidor de aplicativos Sun ONE, o AO Solaris 9 facilita a implementação de serviços escaláveis e confiáveis por demanda, sem nenhum custo adicional. As próximas atualizações do AO contarão com a total integração do servidor de aplicativos Sun ONE, tornando o ambiente operacional Solaris 9 o único ambiente operacional a incluir um servidor de aplicativos J2EE totalmente integrado, de nível comercial, e servidor de diretórios LDAP em seu sistema operacional central.

12.4 Gerenciabilidade

O ambiente operacional Solaris 9 também contribui para a redução de custos de operações com novo dimensionamento e recursos de gerenciamento da mudança, que reduzem o tempo de instalação e os erros. Com recursos como o software Solaris Flash, clientes que usam o Solaris 9, já podem executar em 20 minutos uma operação que antes levava 4 horas com total controle de configuração e alta precisão. Esse recurso representa a redução de horas de trabalho repetitivo para a equipe de TI, proporcionando eficiências ao sistema de gerenciamento. Além disso, o software Resource Manager do Solaris 9 oferece gerenciamento integrado de recursos, o que permite a consolidação de servidores comerciais, sem aquisição de nenhum produto adicional. O software Solaris Volume Manager também contribui para a grande disponibilidade e confiabilidade de dados para os usuários do ambiente operacional Solaris 9.

Boyd Fletcher, analista de sistemas da EG&G, importante empresa de fornecimento de serviços de gerenciamento e técnicos para o governo dos EUA e firmas comerciais, comenta: "Os recursos de gerenciamento de dados do ambiente operacional Solaris 9 oferecem uma plataforma integrada para atender aos requisitos do banco de dados de produção e sistema de arquivos dos clientes da EG&G. A integração do software Solaris Volume Manager com o AO Solaris 9 representa uma economia significativa - US$ 50.000 por site em média. E as melhorias do UFS (Unix File System) e NFS (Network File System) permitem maior rapidez na criação de sistemas de arquivos com melhor desempenho e confiabilidade."

12.5 Segurança

O ambiente operacional Solaris 9 oferece soluções prontas seguras e um conjunto de serviços totalmente integrados para oferecer o nível de segurança necessário aos serviços de rede. "O nível de ataques automáticos sofisticados e de hostilidade na Internet é muito mais elevado do que no passado", diz Graham Lovell, diretor de Produtos de marketing do Solaris, da Sun Microsystems. "Mais do que nunca, os sistemas precisam de segurança em todos os lugares".
Para atender a essas exigências, a nova versão integra vários recursos chave de segurança. O AO Solaris 9 é vendido com um firewall comercial, o software SunScreen[tm] 3.2. Além disso, o software Solaris Secure Shell oferece acesso remoto seguro, criptografado e auditado. Outros recursos de segurança são o LDAP seguro, o servidor Kerberos v5 e o acesso seguro via servidor de portal Sun ONE. Vale repetir que todos esses robustos recursos de segurança fazem parte do AO básico, permitindo que os clientes disponham de aplicativos seguros de ponta a ponta sem nenhum custo adicional.

12.6 Desempenho

As demandas da computação de alto desempenho (HDC) e empresarial exigem ambientes de 64 bits altamente confiáveis e escaláveis. O software do Solaris 9 e os processadores UltraSPARC® III Cu são capazes de fazer os aplicativos trabalharem mais rapidamente sem recompilar, recodificar ou refazer a arquitetura. Alguns dos novos recursos apresentaram as seguintes melhorias de desempenho:

A biblioteca com multithreading aperfeiçoada melhora o OLAP (teste interno Sun do Oracle Express) em até quatro vezes.

O controlador de memória on-chip do UltraSPARC III Cu permite a localização de dados e códigos, otimizada para aumentar o desempenho do sistema em 5 a 40% para servidores de ponta.

O DTLB (Dual Translation Lookaside Buffer) duplo do UltraSPARC III Cu com suporte para páginas grandes melhora os resultados da computação de alto desempenho (HPC) em cerca de três vezes (medido com o padrão SWIM).

O algoritmo de cor Advanced Page aumenta o desempenho do sistema em até 10% para cargas gerais em servidores.

12.7 Disponibilidade

Os recursos da nova versão foram projetados para aumentar a disponibilidade e reduzir o custo total das operações. Entre eles estão a integração avançada com o software Sun Cluster 3.0 e recursos de gerenciamento de configuração. Além disso, a tecnologia e os processos RAS (Remote Access Services) Profile da Sun fornecem as melhores soluções para otimizar as plataformas dos clientes. Estes, aliás, indicam o baixo índice de chamadas de serviços, poucos escalonamentos, baixo tempo de inatividade não programada e aumento de produtividade, elementos que geram economia e maior eficiência.

O Laboratório Nacional de Engenharia e Meio-ambiente de Idaho, onde o Solaris é o ambiente operacional preferido há mais de 11 anos, está convencido da comprovada disponibilidade e estabilidade do AO. "A instalação do Solaris 9 era o próximo passo lógico para obter o máximo em confiabilidade, robustez e recursos do sistema operacional", disse L. Eric Greenwade, cientista do laboratório.

"Na condição de laboratório nacional que fornece soluções científicas e de engenharia para o Departamento de Energia do Governo dos EUA, operamos centenas de sistemas", explica Greenwade. "Dessa forma, não podemos ter muito tempo de inatividade. O AO Solaris 9 é extremamente estável e nos oferece uma plataforma de serviços de Web muito confiável. Até agora, migramos vários aplicativos para Web para o AO Solaris 9. O sistema operacional parece ser totalmente à prova de bala e tivemos poucas paralisações, o que nos ajudará a continuar oferecendo serviços de primeira qualidade aos clientes”.

13 Referências

1) An Overview of the UltraSPARC II Processor. Disponível em: <http://www.sun.com/processors/UltraSPARC-II/ >.

2) An Overview of the UltraSPARC III Processor. Disponível em: <http://www.sun.com/processors/UltraSPARC-III/>.

3) SPARC Open for Business. Disponível em: <http://www.sparc.com/business.html>.

4) SPARC International Features. Disponível em: <http://www.sparc.org/>.

5) Arquitetura de processadores SARC SUN
Leizza Ferreira Rodrigues
Luis Augusto Angelotti Meira
Universidade de Campinas

6) History of SPARC systems – the first decade 1987-1996. Disponível em: <http://www.sparcproductdirectory.com/history.html>.

7) <http://www.sparc.org/history.html>

8) An Overview of the UltraSPARC III Cu Processor. Santa Clara, CA, USA, 2002. Disponível em: <http://www.sun.com/processors/UltraSPARC-III/>.

9) Standard Performance Evaluation Corporation.2000. Disponível em: <http://www.spec.org/>.

10) OnLine Sun information archive. Disponível em: <http://www.sunstuff.org/hardware/systems/sun4/sun4/4-260>.

11) SuperSPARC II User's Manual 1994. Disponível em: <http://www.sun.com/oem/products/manuals/STP1021UG.pdf>.

12) www.sun.com

13) http://home1.swipnet.se/~w-10694/helpers.html

14) http://www.solarisresources.com/

15) CBPF-NT-004/97 - Migração do Solaris 1.x para o Solaris 2.x – Usuários

16) CBPF-NT-004/96 - Introdução a Administração do Sistema Solaris 2