

COMP3320/COMP6464: High Performance Processors

Multiple Register Sets!

Alistair Rendell

See: *High Performance Computing*, Dowd and Severance, Chapter 2
Structured Computer Organization, Tanenbaum, Chapter 8 in Ed 3
SPARC Architecture, Assembly Language Programming and C, Richard Paul
The SPARC Architecture Manual, Version 9 at <http://www.sparc.org/>

5.1 Function Calls

- Historically state of registers stored on the stack prior to function invocation
 - Eg in PeANUt may want to store index register to stack before function call
- Modern computers have special instructions, eg SPARC provides the save instruction:

```
save %sp,-120,%sp
```

- provides a new register window, and decrements stack pointer by 120
 - stack space reserves space for all relevant registers
 - possibly only active registers are saved
- Limited number of arguments are passed via the 8 shared ins(CWP)/outs(CWP-1) registers
 - recall most procedures have few arguments
 - stack pointer is one of these registers

5.2 Registers and Register Windows

Window (CWP - 1)

r[31]
⋮
INS
⋮
r[24]

r[23]
⋮
LOCAL
⋮
r[16]

r[15]
⋮
OUTS
⋮
r[8]

Window (CWP)

r[31]
⋮
INS
⋮
r[24]

r[23]
⋮
LOCAL
⋮
r[16]

r[15]
⋮
OUTS
⋮
r[8]

Window (CWP + 1)

r[31]
⋮
INS
⋮
r[24]

r[23]
⋮
LOCAL
⋮
r[16]

r[15]
⋮
OUTS
⋮
r[8]

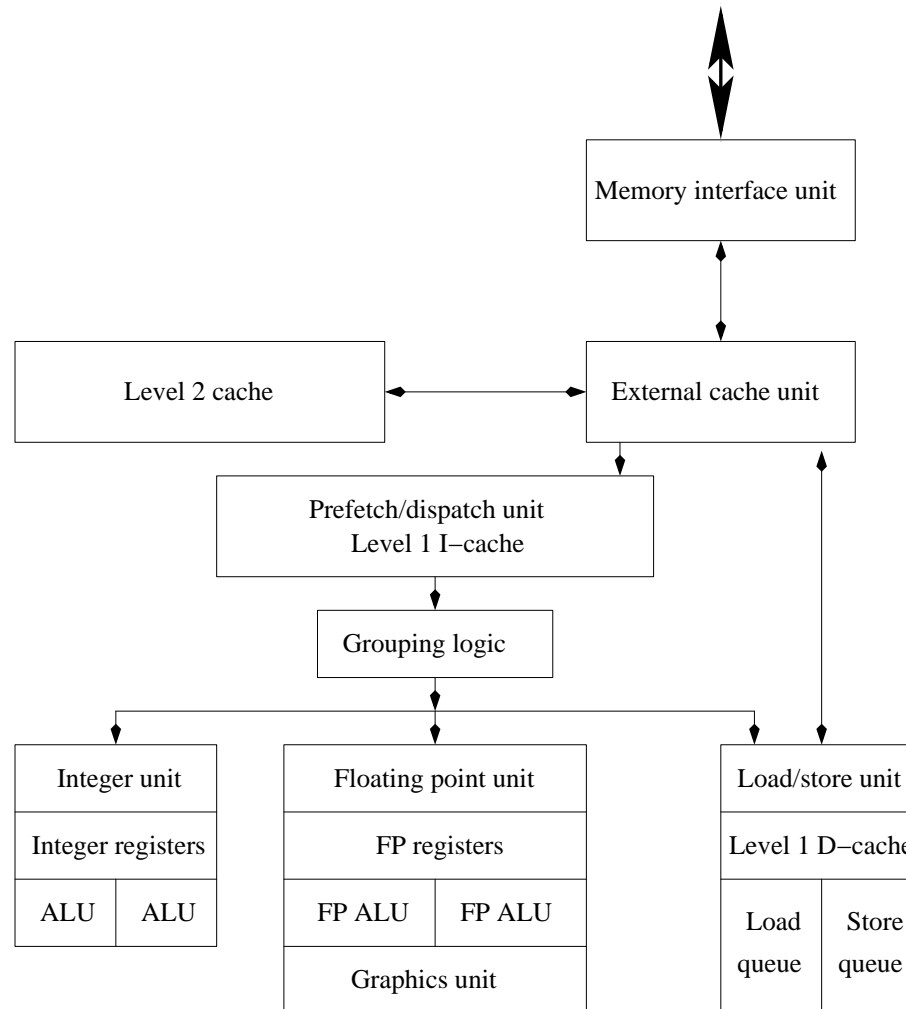
r[7]
⋮
GLOBALS
⋮
r[1]
r[0] 0

63 Bits 0

* 32 64-bit registers are visible at any time

* Which 32 registers is determined via a window register pointer

5.3 UltraSPARC Microarchitecture



5.4 SPARC Assembler #1

- Examine `sum.s` produced via `gcc -S sum.c`

```
int sum(int n1, int n2)
{
    int tmp;

    tmp = n1+n2;
    return tmp;
}
```

```
.section      ".text"
    .align 4
    .global sum
    .type     sum,#function
    .proc     04

sum:
    !#PROLOGUE# 0
    save %sp,-120,%sp
    !#PROLOGUE# 1
    st %i0,[%fp+68]
    st %i1,[%fp+72]
    ld [%fp+68],%o0
    ld [%fp+72],%o1
    add %o0,%o1,%o0
    st %o0,[%fp-20]
    ld [%fp-20],%o0
    mov %o0,%i0
    b .LL1
    nop
.LL1:
    ret
    restore
.LLfe1:
    .size     sum,.LLfe1-sum
    .ident    "GCC: (GNU) 2.8.1"
```

5.5 SPARC Assembler #2

```
int repeat(int n)
{
    int tmp,i;

    tmp = 0;
    for (i=0; i<n; i++)
        tmp = tmp + i;

    return tmp;
}
```

```
st %i0,[%fp+68]
st %g0,[%fp-20]
st %g0,[%fp-24]
```

LL2:

```
ld [%fp-24],%o0
ld [%fp+68],%o1
cmp %o0,%o1
bl .LL5
nop
b .LL3
nop
```

LL5:

```
ld [%fp-20],%o0
ld [%fp-24],%o1
add %o0,%o1,%o0
st %o0,[%fp-20]
```

LL4:

```
ld [%fp-24],%o0
add %o0,1,%o1
st %o1,[%fp-24]
b .LL2
nop
```

LL3:

```
!return value & exit
ld [%fp-20],%o0
mov %o0,%i0
```