BERGISCHE UNIVERSITÄT
GESAMTHOCHSCHULE WUPPERTAL
GAUSS-STRASSE 20
42097 WUPPERTAL
(Korrespondenzanschrift)
42119 WUPPERTAL
(Lieferanschrift)
TELEX 8 592 262 bughw
TELEFAX (0202) 439-2901
TELEFON (0202) 439-1

Fachbereich 7

MATHEMATIK

Prof. Dr. Hans-Jürgen Buhl
Praktische Informatik / Numerik

e-mail: Juergen.Buhl@math.uni-wuppertal.de

Grundlagen der Rechnerarchitektur – Neuere Entwicklungen in der Informationstechnologie (Einführung in die Wirtschaftsinformatik II)

SS 2003 – Übungsblatt 9

18. Juli 2003
Ausgabe: 7. Juli 2003

Aufgabe 1. *Befehlsklassen der SPARC RISC-CPU*

Welche (Maschinen-)Befehlsklassen sind SPARC-spezifisch, kommen zum Beispiel beim Intel Pentium nicht vor?

Aufgabe 2. *SPARC- und x86-Maschinencode*

Das folgende kleine Programm `first.c`

```
#include        <stdio.h>
#include        <math.h>

double f(double x);

int main(){

  int i;

  printf(" x       |   f(x)\n");
  printf("_____________________\n\n");

  for (i = 0; i <=20; i++)
    printf("%f   %f \n", i/10.0, f(i/10.0));
}

double f(double x){

  double arg;
```

```
  arg= 2 * M_PI * x + 5.0;
  return sin(arg);

}
```

kann mittels `cc -S first.c` in das entsprechende Maschinenprogramm Ihres Computers übersetzt werden.

Im SPARC-Code sieht das folgendermaßen aus:

```
.section ".text",#alloc,#execinstr
.align 8
.skip 16

! block 0

.global main
.type main,2
main:
save %sp,-120,%sp

! block 1
.L171:

! File first.c:
!    1 #include <stdio.h>
!    2 #include <math.h>
!    3
!    4 double f(double x);
!    5
!    6 int main(){
!    7
!    8   int i;
!    9
!   10   printf(" x       |   f(x)\n");

sethi %hi(.L172),%o0
or %o0,%lo(.L172),%o0
call printf
nop

!   11   printf("____________________\n\n");

sethi %hi(.L173),%o0
or %o0,%lo(.L173),%o0
call printf
nop

!   13   for (i = 0; i <=20; i++)
```

```
st %g0,[%fp-8]
ld [%fp-8],%l0
cmp %l0,20
bg .L176
nop

! block 2
.L177:
.L174:

!   14     printf("%f   %f \n", i/10.0, f(i/10.0));

sethi %hi(.L178),%l1
or %l1,%lo(.L178),%l1
ld [%fp-8],%l0
st %l0,[%fp-12]
ld [%fp-12],%f2
fitod %f2,%f4
sethi %hi(.L_cseg0),%l0
ldd [%l0+%lo(.L_cseg0)],%f2
fdivd %f4,%f2,%f30
std %f30,[%fp-24]
ld [%fp-8],%l0
st %l0,[%fp-12]
ld [%fp-12],%f2
fitod %f2,%f4
sethi %hi(.L_cseg0),%l0
ldd [%l0+%lo(.L_cseg0)],%f2
fdivd %f4,%f2,%f2
std %f2,[%sp+88]
call f
ldd [%sp+88],%o0
fmovs %f1, %f3
fmovs %f0,%f2
mov %l1,%o0
ldd [%fp-24],%f30
std %f30,[%sp+88]
ld [%sp+88],%o1
ld [%sp+92],%o2
std %f2,[%sp+88]
ld [%sp+88],%o3
call printf
ld [%sp+92],%o4
ld [%fp-8],%l0
add %l0,1,%l0
st %l0,[%fp-8]
ld [%fp-8],%l0
cmp %l0,20
```

```
ble .L174
nop

! block 3
.L179:
.L176:
.L170:
jmp %i7+8
restore
.size main,(.-main)
.align 8
.align 8
.skip 16

! block 0

.global f
.type f,2
f:
save %sp,-120,%sp
st %i1,[%fp+72]
st %i0,[%fp+68]

! block 1
.L182:
ld [%fp+68],%f6
ld [%fp+72],%f7
std %f6,[%fp-16]

! File first.c:
!   15 }
!   16
!   17 double f(double x){
!   18
!   19   double arg;
!   20
!   21   arg= 2 * M_PI * x + 5.0;

sethi %hi(.L_cseg1),%l0
ldd [%l0+%lo(.L_cseg1)],%f4
sethi %hi(.L_cseg2),%l0
ldd [%l0+%lo(.L_cseg2)],%f2
fmuld %f4,%f2,%f2
fmuld %f2,%f6,%f4
sethi %hi(.L_cseg3),%l0
ldd [%l0+%lo(.L_cseg3)],%f2
faddd %f4,%f2,%f2
std %f2,[%fp-24]
```

```
!   22   return sin(arg);

std %f2,[%sp+88]
call sin
ldd [%sp+88],%o0
fmovs %f1, %f3
fmovs %f0,%f2
std %f2,[%fp-8]
fmovs %f3, %f1
fmovs %f2,%f0
jmp %i7+8
restore

! block 2
.L181:
fmovs %f3, %f1
fmovs %f2,%f0
jmp %i7+8
restore
.size f,(.-f)
.align 8

.section ".rodata1",#alloc
.align 4
.L172:
.ascii " x       |   f(x)\n\000"
.skip 1
.type .L172,#object
.size .L172,20
.align 4
.L173:
.ascii "____________________\n\n\000"
.skip 1
.type .L173,#object
.size .L173,24
.align 4
.L178:
.ascii "%f   %f \n\000"
.type .L178,#object
.size .L178,10

.section ".rodata",#alloc
.align 8
.L_cseg0:
.word 0x40240000,0x0
.type .L_cseg0,#object
.size .L_cseg0,8
.align 8
.L_cseg1:
```

```
.word 0x40000000,0x0
.type .L_cseg1,#object
.size .L_cseg1,8
.align 8
.L_cseg2:
.word 0x400921fb,0x54442d18
.type .L_cseg2,#object
.size .L_cseg2,8
.align 8
.L_cseg3:
.word 0x40140000,0x0
.type .L_cseg3,#object
.size .L_cseg3,8

.file "first.c"
.xstabs ".stab.index","Xa ; V=3.1 ; R=WorkShop Compilers 5.0 98/12/15 C 5.0
.xstabs ".stab.index","/home/buhl/prg-sprachen; /opt/share/SUNWspro/bin/../
.xstabs ".stab.index","main",42,0,0,0
.ident "@(#)stdio.h 1.69 98/07/13 SMI"
.ident "@(#)feature_tests.h 1.17 97/12/04 SMI"
.ident "@(#)isa_defs.h 1.16 99/05/25 SMI"
.ident "@(#)va_list.h 1.11 97/11/22 SMI"
.ident "@(#)stdio_tag.h 1.3 98/04/20 SMI"
.ident "@(#)stdio_impl.h 1.8 99/06/30 SMI"
.ident "@(#)math.h 2.7 98/01/27"
.ident "@(#)floatingpoint.h 2.4 94/06/09"
.ident "@(#)ieeefp.h 2.7 94/11/09"
.ident "acomp: WorkShop Compilers 5.0 98/12/15 C 5.0"

.global __fsr_init_value
__fsr_init_value = 0x0
```

Erläutern Sie die Maschinenanweisungen, die zum Aufruf `printf(" x | f(x)\n");` gehören.

Beantworten Sie dieselbe Frage im Code eines Intel-x86-Programms:

```
.file "first.c"
.section .rodata
.LC0:
.string " x       |   f(x)\n"
.LC1:
.string "____________________\n\n"
.LC3:
.string "%f   %f \n"
.align 8
.LC2:
.long 0
.long 1076101120
```

```
.text
.globl main
.type main, @function
main:
pushl %ebp
movl %esp, %ebp
subl $8, %esp
andl $-16, %esp
movl $0, %eax
subl %eax, %esp
subl $12, %esp
pushl $.LC0
call printf
addl $16, %esp
subl $12, %esp
pushl $.LC1
call printf
addl $16, %esp
movl $0, -4(%ebp)
.L2:
cmpl $20, -4(%ebp)
jle .L5
jmp .L3
.L5:
subl $12, %esp
subl $12, %esp
fildl -4(%ebp)
fldl .LC2
fdivrp %st, %st(1)
leal -8(%esp), %esp
fstpl (%esp)
call f
addl $20, %esp
leal -8(%esp), %esp
fstpl (%esp)
fildl -4(%ebp)
fldl .LC2
fdivrp %st, %st(1)
leal -8(%esp), %esp
fstpl (%esp)
pushl $.LC3
call printf
addl $32, %esp
leal -4(%ebp), %eax
incl (%eax)
jmp .L2
.L3:
leave
ret
```

```
.size main, .-main
.section .rodata
.align 8
.LC5:
.long 1413754136
.long 1075388923
.align 8
.LC6:
.long 0
.long 1075052544
.text
.globl f
.type f, @function
f:
pushl %ebp
movl %esp, %ebp
subl $24, %esp
movl 8(%ebp), %eax
movl 12(%ebp), %edx
movl %eax, -8(%ebp)
movl %edx, -4(%ebp)
fldl -8(%ebp)
fldl .LC5
fmulp %st, %st(1)
fldl .LC6
faddp %st, %st(1)
fstpl -16(%ebp)
subl $8, %esp
pushl -12(%ebp)
pushl -16(%ebp)
call sin
addl $16, %esp
leave
ret
.size f, .-f
.ident "GCC: (GNU) 3.3 20030226 (prerelease) (SuSE Linux)"
```

Zeigen Sie ebenfalls alle Maschinenbefehle, die zur Schleife `for (i = 0; i <=20; i++)` gehören (Zählerinitialisierung, Schleifenabbruchsbedingungsüberprüfung, Inkrementierung des Zählers).

Aufgabe 3. *SIMD in SPARC-CPUs*

Informieren Sie sich über die VIS-Instruktionen der SPARC-Rechner:

http://www.sun.com/processors/vis/

Welche Zwecke erfüllt diese Instruktionsklasse? Erläutern Sie den Befehl `PDIST` (Seite 17 des Handbuchs »`The VIS Instruction Set`«,

http://www.sun.com/processors/whitepapers/vis_wp_external.pdf

) in eigenen Worten.

Aufgabe 4. *SIMD beim Intel Pentium*

Informieren Sie sich im »Intel Pentium 4 Users Manual«

ftp://download.intel.com/design/pentium4/manuals/24547012.pdf

über die zu VIS analoge Befehlsklasse beim Pentium 4 (SSE und SSE2). Gibt es hier einen zu `PDIST` äquivalenten Befehl. Überlegen Sie sich bei Nichtexistenz eine Befehlsfolge, die denselben Zweck erfüllen kann.