Abgabe von Bruno Stendal, Martin Baer, Lukas Gewinner und Christian Schäfer
6. Aufgabenblatt zum Kurs

TI2: Rechnerarchitektur

von Bernadette Keßler

bis Freitag, den 16.12.2022, 10:15 Uhr.

Fließbandverarbeitung

Sorgen Sie dafür dass die folgende Befehlsfolge aus Pseudoinstructions in den jeweiligen Pipelines konfliktfrei ausgeführt wird.
Programmcodeunterteilung:

1. add r0, r1, r2
2. add r1, r5, r0
3. mov [rsp+8], r5
4. or r0, r5, r4
5. mov r3, [rsp+24]
6. and r1, r0, r3
7. add r0, r1, r3
8. add r0, r0, 0x341D
9. add r0, r0, 0x52F6

Einfache Pipeline
Gehen Sie hier von einer einfachen 5-stufigen Pipeline aus: Befehl holen (IF), Befehl dekodieren (ID), Operanden holen (OF), Ausführung (EX), Rückspeichern (WB). Weiterhin liegt eine reine Load/Store-Architektur ohne architekturelle Beschleunigungsmaßnahmen (z.B. Forwarding, Reordering etc.) oder Hardware zur Erkennung von Hemmnissen vor. Operanden können erst dann aus Registern geholt werden, nachdem sie zurück gespeichert wurden.

Step	IF	ID	OF	EX	WB
1.	1	NOP	NOP	NOP	NOP
2.	NOP	1	NOP	NOP	NOP
3.	NOP	NOP	1	NOP	NOP
4.	2	NOP	NOP	1	NOP
5.	NOP	2	NOP	NOP	1
6.	NOP	NOP	2	NOP	NOP
7.	3	NOP	NOP	2	NOP
8.	NOP	3	NOP	NOP	2
9.	NOP	NOP	3	NOP	NOP
10.	4	NOP	NOP	3	NOP
11.	NOP	4	NOP	NOP	3
12.	NOP	NOP	4	NOP	NOP
13.	5	NOP	NOP	4	NOP
14.	NOP	5	NOP	NOP	4
15.	NOP	NOP	5	NOP	NOP
16.	6	NOP	NOP	5	NOP
17.	NOP	6	NOP	NOP	5
18.	NOP	NOP	6	NOP	NOP
19.	7	NOP	NOP	6	NOP
20.	NOP	7	NOP	NOP	6
21.	NOP	NOP	7	NOP	NOP
22.	8	NOP	NOP	7	NOP
23.	NOP	8	NOP	NOP	7
24.	NOP	NOP	8	NOP	NOP
25.	9	NOP	NOP	8	NOP
26.	NOP	9	NOP	NOP	8
27.	NOP	NOP	9	NOP	NOP
28.	NOP	NOP	NOP	9	NOP
29.	NOP	NOP	NOP	NOP	9
30.			Ende.		

Verbesserte Pipeline
Gehen Sie jetzt davon aus, dass die Pipeline aus dem vorherigen Aufgabenteil die in der Vorlesung kennengelernten Forwarding-Varianten/Shortcuts verwendet.

Step	IF	ID	OF	EX	WB
1.	1	NOP	NOP	NOP	NOP
2.	2	1	NOP	NOP	NOP
3.	3	2	1	NOP	NOP
4.	3	2	NOP	1	NOP
5.	NOP	3	2	NOP	1
6.	4	NOP	3	2	NOP
7.	5	4	NOP	3	2
8.	6	5	4	NOP	3
9.	NOP	6	5	4	NOP
10.	7	NOP	6	5	4
11.	NOP	7	NOP	6	5
12.	8	NOP	7	NOP	6
13.	NOP	8	NOP	7	NOP
14.	9	NOP	8	NOP	7
15.	NOP	9	NOP	8	NOP
16.	NOP	NOP	9	NOP	8
17.	NOP	NOP	NOP	9	NOP
18.	NOP	NOP	NOP	NOP	9
19.			Ende.		

Arithmetik

Integer-Arithmetik
Implementieren Sie eine Funktion, die den ganzzahliegen Anteil der Formel berechnet.
Fließkomma-Arithmetik
Implementieren Sie die obige Formel für Fließkommazahlen.

```
;Bruno Stendal, Martin Baer, Lukas Gewinner, Christian Schaefer
GLOBAL formula_int, formula_flt

SECTION .data;

null:   dq  0.0    ;
acht:   dq  8.0    ;
vier:   dq  4.0    ;
zwei:   dq  2.0    ;
drei:   dq  3.0    ;



SECTION .text;

formula_int:
    .start:
        mov rax, 0; macht rax = 0
```

```
19          ;a = rdi
20          ;b = rsi
21          ;c = rdx => ueberlauf von mul und div => r11
22          mov r11, rdx;
23          ;d = rcx
24          ;e = r8
25          ;f = r9
26          ;g = [rsp+8]
27          ;h = [rsp+16]
28
29          ;  berklammer   eins
30          ;klammer ab
31          add rdi, rsi;
32          ;klammer cd
33          sub r11, rcx;
34
35          ;multiplizieren ab und cd
36          mov rax, rdi;
37          mul r11;
38          mov rdi, rax;
39
40          ;  berklammer   zwei
41          ;e*8
42          mov rax, 8;
43          mul r8;
44          mov r8, rax;
45          ;f*4
46          mov rax, 4;
47          mul r9;
48          mov r9, rax;
49          ;g/2
50          mov rax, [rsp+8];
51          cmp rax, 0;
52          jl .gInvert;
53          jg .gNormal;
54          je .end;
55
56      .gInvert:
57          neg rax;
58          mov r10, 2;
59          div r10;
60          neg rax;
61          jmp .gContiue;
62
63      .gNormal:
64          mov r10, 2;
65          div r10;
```

```
66          jmp .gContiue;
67
68      .gContiue:
69          mov [rsp+8], rax;
70          ;h/4
71          mov rax, 0;
72          mov r10, 0;
73          mov rax, [rsp+16];
74          cmp rax, 0;
75          jl .hInvert;
76          jg .hNormal;
77          je .end;
78
79      .hInvert:
80          neg rax;
81          mov r10, 4;
82          div r10;
83          neg rax;
84          jmp .hContinue;
85
86      .hNormal:
87          mov r10, 4;
88          div r10;
89          jmp .hContinue;
90
91      .hContinue:
92          mov [rsp+16], rax;
93
94          ;  berklammer   zwei aufaddieren
95          add r8, r9;
96          sub r8, [rsp+8];
97          add r8, [rsp+16];
98
99          ;  berklammern   multiplizieren
100         mov rax, rdi;
101         mul r8;
102         mov rdi, rax;
103         mov rax, 0;
104
105         ;division durch 3
106         mov rax, 0;
107         mov rdx, 0;
108         mov r10, 0;
109
110         mov rax, rdi;
111
112         cmp rax, 0;
```

```
113         jl .dreiInvert;
114         jg .dreiNormal;
115         je .end;
116
117
118     .dreiNormal:
119         mov r10, 3;
120         div r10;
121         jmp .end;
122
123     .dreiInvert
124         neg rax;
125         mov r10, 3;
126         div r10;
127         neg rax;
128         jmp .end;
129
130     .end;
131         ret;
132
133
134 formula_flt:
135     ;a = xmm0
136     ;b = xmm1
137     ;c = xmm2
138     ;d = xmm3
139     ;e = xmm4
140     ;f = xmm5
141     ;g = xmm6
142     ;h = xmm7
143     addsd xmm0, xmm1;
144     movsd xmm1, xmm0;               xmm1 = a+b
145     movsd xmm0, [rel null];
146     subsd xmm2, xmm3;               xmm2 = c-d
147
148
149     mulsd xmm4, [rel acht];
150     mulsd xmm5, [rel vier];
151     divsd xmm6, [rel zwei];
152     divsd xmm7, [rel vier];
153
154     mulsd xmm1, xmm2;               xmm1 = (a+b)*(c-d)
155     movsd xmm0, xmm1;
156
157     addsd xmm4, xmm5;               zweite   berklammer
158     subsd xmm4, xmm6;
159     addsd xmm4, xmm7;
```

```
160
161     mulsd xmm1, xmm4;                multiplizieren   berklammern
162     divsd xmm1, [rel drei];          division mit drei
163
164     movsd xmm0, xmm1;
165
166     ret;
```

Teil 1

- g und h liegen in [rsp+8] und [rsp+16]
- Die division von negativen Zahlen ist nicht möglich daher müssen wir das Vorzeichen vor und nach der Divsion umkehren
- 2^{31}, nicht 32 da wir ein signed Integer haben, hat einen maximalen Zahlenwert von 2.147.483.648 was wenn ein Paramter über 2 Milliarden ist überschritten werden würde und einen Overflow produzieren würde.

Teil 2

- Die xmm Register von xmm0 bis xmm15
- addsd, divsd, mulsd und subsd, wobei das s am ende für scalar steht und das d für double precision also 64bit steht.
- Alle xmm Register gelten als Volatile und werden bei einem Funktionsaufrum genullt.