Freie Universität Berlin

Übung 7

Rechnerarchitektur
Wintersemester 17/18
Prof. Dr.-Ing. Jochen Schiller

Ausgabe 01.12.2017
Abgabe 15.12.2017, 10:15 (s.t.)

Bitte beachten Sie die allgemeinen Hinweise auf Übungszettel 1

Aufgabe 1: Fließbandverarbeitung

a) Einfache Pipeline

Gehen Sie im Folgenden 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.

Sorgen Sie dafür dass die folgende Befehlsfolge aus Pseudoinstructions konfliktfrei ausgeführt wird:

```
ADD r0, r1, r2
SUB r1, r5, r0
MOV [rsp+8], r5
OR r0, r5, r4
MOV r3, [rsp+24]
AND r1, r0, r3
ADD r0, r1, r3
ADD r0, r0, 0x341D
ADD r0, r0, 0x52F6
```

Hinweis: `ADD a, b, c` *entspricht* `a=b+c`

b) Verbesserte Pipeline

Gehen Sie jetzt davon aus, dass die Pipeline aus dem vorherigen Aufgabenteil Forwarding/Shortcuts verwendet. Sorgen Sie wieder dafür, dass die Befehlsfolge aus dem vorherigen Aufgabenteil konfliktfrei ausgeführt wird.

Aufgabe 2: Alte Klausuraufgabe

Hinweise:

- Es handelt sich um eine Aufgabe, wie sie auch in der Klausur gestellt sein könnte.
- Sie sollten inzwischen in der Lage sein, die Aufgabe selbstständig zu bearbeiten.
- Die Aufgabe ist nicht ganz leicht, aber eine gute Gelegenheit Ihre Herangehensweise zu üben.
- Wichtig: Lesen Sie sich zuerst die Aufgabenstellung vollständig durch. Wahrscheinlich müssen Sie nicht das ganze Programm verstehen um die Fragen zu beantworten.
- Wenn Sie diese und ähnliche Aufgaben unter 15 Minuten vollständig lösen können (Fragen 1-4), sind Sie für die Klausur sehr gut gerüstet!

Gegeben sei folgende NASM-Funktion (Parameter stehen in rdi und rsi, der Rückgabewert steht in rax):

```
01  GLOBAL function

02  function:
03      MOV r8, 1

04  _loop:
05      CMP r8, rsi
06      JGE _endloop

07      MOV al, [rdi+r8]
08      MOV r9, r8
09      MOV r10, r8

10  _innerloop:
11      DEC r9
12      CMP r9, 0
13      JL  _endinnerloop

14      MOV bl, [rdi+r9]
15      CMP al, bl
16      JL  _endinnerloop

17      MOV [rdi+r9], al
18      MOV [rdi+r10], bl
19      DEC r10
20      JMP _innerloop

21  _endinnerloop:
22      INC r8
23      JMP _loop

24  _endloop:
25      MOV rax, rdi
26      RET
```

Aufgerufen wird diese Funktion folgendermaßen:

```
int8_t array[] = "World";
function(array, 5);
```

Sie können diese ASCII-Tabelle zur Hilfe nehmen:

Char	W	d	l	o	r
ASCII (Dezimal)	87	100	108	111	114

Beantworten Sie folgende Fragen:

1. Geben Sie den Inhalt des Arrays nach dem ersten Schleifendurchlauf von *_loop* an.
2. Wie oft wird die innere Schleife (*_innerloop*) insgesamt ausgeführt?
3. Welchen Rückgabewert gibt die Funktion zurück?
4. Wie sieht der Inhalt der Variable *array* nach dem Funktionsaufruf aus?
5. (Optional:) Was tut das Programm?