64-041 Übung Rechnerstrukturen und Betriebssysteme

Aufgabenblatt 12 Ausgabe: 03.02., Abgabe: 10.02. 24:00

Gruppe	
Name(n)	Matrikelnummer(n)

Aufgabe 12.1 (Punkte 6·4)

x86-Adressierung: Angenommen, die folgenden Werte sind in den angegebenen Registern bzw. Speicheradressen gespeichert. Adressen sind verkürzt dargestellt und von den Werten sind jeweils auch nur die unteren 32-bit angegeben:

Register	Wert
`%rax`	`0x00000108`
`%rcx`	`0x0000010c`
`%rdx`	`0x00000008`
`%rbx`	`0x00000100`

Adresse	Wert
`0x100`	`0x000000ba`
`0x108`	`0x0000cafe`
`0x110`	`0x000000bf`
`0x118`	`0x00009876`

Überlegen Sie sich, welche Speicheradressen bzw. Register als Ziel der folgenden Befehle ausgewählt werden und welche Resultatwerte sich aus den Befehlen ergeben:

Befehl	Ziel (Adresse/Register)	Wert
`addq %rcx, (%rbx)`		
`subq %rdx, 0x8(%rax)`		
`salq $16, %rax`		
`xorq %rcx, 16(%rax)`		
`decq 4(%rcx)`		
`subq %rdx, %rcx`		

Zur Erinnerung: für den gnu-Assembler gilt

- der Zieloperand steht rechts
- Registerzugriffe werden direkt ausgedrückt
- eine runde Klammer um ein Register bedeutet einen Speicherzugriff, ggf. mit Immediate-Offset und Index: $\langle imm \rangle$`(`$\langle Rb \rangle$`,`$\langle Ri \rangle$`,`$\langle s \rangle$`)` → `MEM[`$\langle Rb \rangle$`+`$\langle s \rangle$`*`$\langle Ri \rangle$`+`$\langle imm \rangle$`]`

⇒ Beispiel: Befehl `addq %rdx, 24(%rbx)`
Operation `MEM[0x00000118] := MEM[0x00000118]+8 = 0x0000987e`

Aufgabe 12.2 (Punkte 7·3)

Flags: Bei dieser Aufgabe sollen Sie sich überlegen, welche Flags (Carry, bzw. Overflow) durch arithmetische Operationen in der CPU gesetzt werden. Kreuzen Sie die Flags an und begründen Sie Ihre Entscheidung kurz (jeweils ein Satz).

Operation	Inhalt `%eax`	Inhalt `%ebx`	`CF`	`OF`
`addl %eax, %ebx`	`0x00000001`	`0x00000002`		
`addl %eax, %ebx`	`0x00000001`	`0xFFFFFFFF`		
`addl %eax, %ebx`	`0x00000002`	`0x7FFFFFFF`		
`addl %eax, %ebx`	`0x80000000`	`0xFFFFFFFE`		
`subl %eax, %ebx`	`0x00000002`	`0x00000001`		
`subl %eax, %ebx`	`0x00000001`	`0x00000002`		
`subl %eax, %ebx`	`0x00000001`	`0x80000000`		

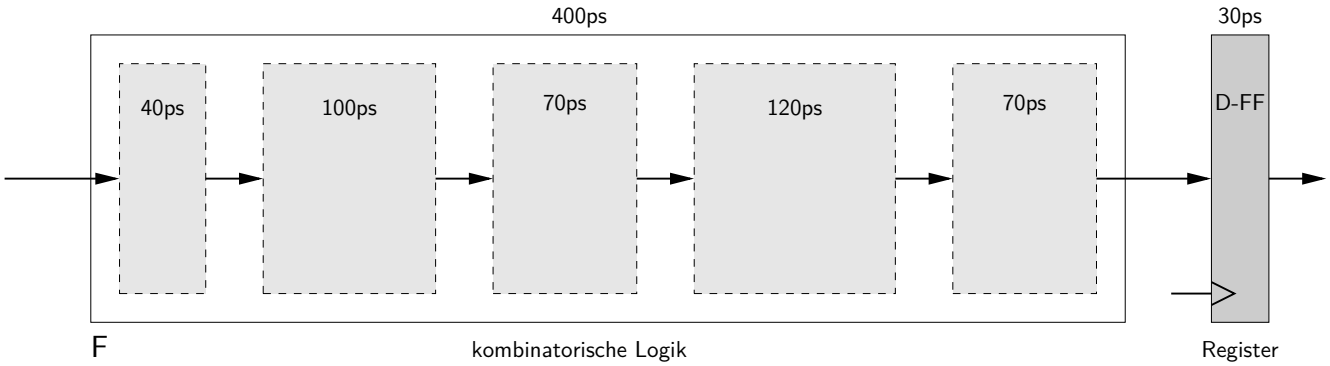
Anmerkung: um nicht mit 64-bit Zahlen zu rechnen, sind die Register, deren Inhalte und die Operationen jeweils 32-bit breit. Bei der x86-Architektur werden die Register `%e..` statt `%r..` benutzt (s. Folie 879) und die Assemblerbefehle sind `addl` statt `addq` (*long* statt *quad*).

Zur Erinnerung: das Carry-Flag wird bei einer arithmetischen Operation gesetzt, wenn sich in der höchstwertigsten Stelle ein Übertrag ergibt. Das Overflow-Flag wird gesetzt, wenn das Ergebnis der Operation das „falsche“ Vorzeichen hat, beispielsweise wenn die Addition zweier positiver Zahlen ein negatives Ergebnis liefert.

Aufgabe 12.3 (Punkte 4·5)

x86-Prozeduraufrufe und Stack: Der Stack ist für die Ausführung von Funktionen, bzw. Prozeduren von zentraler Bedeutung. In der Vorlesung wurden 4 Befehle vorgestellt, die den Stackpointer `%rsp` indirekt verändern, d.h. ohne dass `%rsp` als Argument in dem Befehl vorkommt. Zählen Sie die Befehle auf, jeweils mit einer kurzen Beschreibung was sie bewirken.

Aufgabe 12.4 (Punkte 5+5+5)



Pipelining: Gegeben sei die oben gezeigte Funktionseinheit. Bei einem Signalwechsel am Eingang des kombinatorischen Logikblockes dauert es 400 *ps*, bis der richtige Funktionswert am Ausgang der Funktion *F* erscheint. Für das Register soll der Einfachheit halber lediglich eine Zeitbedingung eingeführt werden: die Zeitdauer zwischen Taktvorderflanke und Ausgabe des neuen Werts am Datenausgang sei 30 *ps*.

(a) Geben Sie die Latenzzeit und den Durchsatz der Schaltung an.

(b) Der kombinatorische Logikblock der obigen Schaltung lässt sich, an den angegebenen Stellen in kleinere Teilfunktionen aufteilen, wobei die Reihenfolge beizubehalten ist. Zeigen Sie, wie durch Einführung von Pipelineregistern (mit gleichen Zeitbedingungen wie das vorhandene Register) der Durchsatz maximiert werden kann.

Wir erlauben zunächst nur ein einziges Pipelineregister, also eine zweistufige Pipeline. An welcher Stelle muss das Pipelineregister für maximalen Durchsatz platziert werden? Geben Sie auch für diese Konfiguration Latenzzeit und Durchsatz an.

(c) Welche Werte ergeben sich, wenn an allen vier möglichen Stellen Pipelineregister eingefügt werden?

Aufgabe 12.5 (Punkte 10·2)

Zwar fehlen noch Vorlesungsinhalte aus „Kapitel 14 – Rechnerarchitektur II“ und „Kapitel 15 – Betriebssysteme“, mit dieser Aufgabe soll aber schon einmal Ihr „Verständnis“ zu den Inhalten der Vorlesung mit Multiple-Choice Fragen überprüft werden. Dazu sollen Sie für die nachfolgenden Aussagen aus verschiedenen Themenbereichen jeweils ankreuzen, ob diese stimmen oder falsch sind.

(a) Im 2-Komplement wird mit dem Vektor „10…01“ die -1 dargestellt.

☐ stimmt ☐ stimmt nicht

(b) Der Morsecode erfüllt die Fano-Bedingung nicht.

☐ stimmt ☐ stimmt nicht

(c) Durch Hinzufügen eines Paritätsbits wird die Fehlerkorrektur von 1-bit Fehlern möglich.

☐ stimmt ☐ stimmt nicht

(d) Ein Mealy-Automat kann immer schneller getaktet werden als ein Moore-Automat.

☐ stimmt ☐ stimmt nicht

(e) Durch *Alignment* der Daten im Speicher wird der Zugriff schneller.

☐ stimmt ☐ stimmt nicht

(f) Cache Speicher ist aus speziellen DRAM Bausteinen aufgebaut.

☐ stimmt ☐ stimmt nicht

(g) Bei einer CISC-Architektur dürfen die meisten ALU-Befehle ihre Operanden aus dem Hauptspeicher laden.

☐ stimmt ☐ stimmt nicht

(h) Beim x86-Assembler wird der Stack benötigt, um Unterprogramme aufzurufen.

☐ stimmt ☐ stimmt nicht

(i) Bei der Programmierung von Schleifen muss der Programmierer auf RISC-Rechnern wegen der zusätzlichen Takte zum Füllen der Pipeline die Laufvariablen anpassen.

☐ stimmt ☐ stimmt nicht

(j) Pipelining ist mit einer von-Neumann Architektur nicht möglich.

☐ stimmt ☐ stimmt nicht