Aufgabenblatt 11

Ausgabe 11/01/2010, Abgabe bis 18/01/2010 12:00

Name(n):
Matrikelnummer(n):
Übungsgruppe:

Aufgabe 11.1 x86-Assembler (5+5+5+5+5+5 Punkte)

Angenommen, die folgenden Werte sind in den angegebenen Registern bzw. Speicheradressen gespeichert:

Register	Wert
`%eax`	`0x00000100`
`%ecx`	`0x00000001`
`%edx`	`0x0000000C`

Adresse	Wert
`0x100`	`0x0000CAFE`
`0x104`	`0x000000AC`
`0x108`	`0x00000013`
`0x10c`	`0x00098700`

Überlegen Sie sich, welche Adressen bzw. Register als Ziel der folgenden Befehle ausgewählt werden und welche Resultatwerte sich ergeben:

Befehl	Ziel (Adresse/Register)	Wert
`addl %ecx, (%eax)`		
`subl %edx,4(%eax)`		
`imull $16,(%eax,%edx)`		
`incl 8(%eax)`		
`decl %ecx`		
`subl %edx,%eax`		

Hinweis:
Beim gnu-Assembler steht der Zieloperand rechts, und eine runde Klammer um ein Register bedeutet einen Speicherzugriff auf die entsprechende Adresse, ggf. mit dem vor der Klammer notierten Byte-Offset. Zum Beispiel bewirkt
der Befehl `addl %ecx,12(%eax)`
die Operation: `MEM[0x0000010c] = 0x00098701`
Sie können die Befehle natürlich auch im Assembler und Debugger direkt ausprobieren.

Aufgabe 11.2 x86-Assembler (5+15 Punkte)

a) Wie kann man den Inhalt eines Registers auf Null setzen, wenn dafür kein separater Befehl zur Verfügung steht? Geben Sie x86-Beispielcode an, der ohne Immediate-Operand auskommt.

b) Wie kann man die Inhalte von zwei Registern vertauschen, ohne ein zusätzliches Register oder eine zusätzliche Speicherstelle zu verwenden? Geben Sie als Beispiel den x86 Assemblercode an, um die Werte in den Registern %eax und %edx zu vertauschen.

Hinweis:
Denken Sie auch über die XOR-Operation nach. Der x86 Befehl dafür lautet `xorl src, dest`.

Aufgabe 11.3 x86-Assembler (20 Punkte)

Es gibt keinen x86-Assemblerbefehl, der es erlaubt, den Programmzähler `%eip` direkt auszulesen. Schreiben Sie ein kurzes Assemblerprogramm, das den Programmzähler in das Register `%eax` kopiert.
Hinweis: Sie dürfen den Stack zur Zwischenspeicherung verwenden.

Aufgabe 11.4 x86-Assembler (30 Punkte)

Eine klassische Aufgabe zur Demonstration einfacher numerischer Operationen ist die Umrechnung zwischen Grad Celsius C und Grad Fahrenheit F nach der Formel $C = (F - 32) * (5/9)$.

Da im bisher eingeführten Befehlssatz für unseren x86-Prozessor noch kein Befehl für die Division enthalten ist, nähern wie den Umrechnungsfaktor $5/9$ durch den Wert $5/9 \approx 142/256$ an, der sich zum Beispiel mit einer Multiplikation (`imull src dest`) und einem Rechts-Shift (`sarl` bzw. `shrl` für arithmetisches und logisches shift-right) effizient umsetzen lässt.

Schreiben Sie x86-Assemblercode für eine Funktion `int celsius( int fahrenheit )`, die ihr Argument (in Grad Fahrenheit), wie in der Vorlesung erläutert, auf dem Stack übergeben bekommt und ihren Rückgabewert entsprechend der Konvention im Register %eax hinterläßt. Nach Ausführung der Funktion sollen die relevanten Datenregister wieder ihren vorherigen Wert enthalten. Bedenken Sie dabei, daß laut Konvention die Register %eax, %edx und %ecx als “caller save” klassifiziert sind. Die Inhalte der für die Berechnung benötigten Register müssen also von der Funktion teilweise ebenfalls auf den Stack gerettet und am Ende wiederhergestellt werden.