UNIVERSITÄT SALZBURG

Rechnerarchitektur

SS 2020

Übungszettel 1

1. Schreiben Sie ein Programmstück für die Stackmaschine, das $A^2 - B^2$ berechnet und das Ergebnis in die Variable C speichert. Es stehen die Befehlscodes `add` (2 Taktzyklen), `sub` (2 Taktzyklen) und mult (8 Taktzyklen) zur Verfügung. Sie sollten dabei mit *einer* Multiplikation auskommen.

   (a) Wieviele Befehle und Taktzyklen benötigt Ihr Programm? Wieviele Bytes (Instruktionen und Daten) werden beim Programmablauf vom Speicher in die Maschine transferiert?

   (b) Es stehen nun die zusätzlichen Befehle `dup` (dupliziert das oberste Stackelement, 1 Taktzyklus) und `exc` (vertauscht die beiden obersten Stackelemente, 1 Taktzyklus) zur Verfügung.

   (c) Ermitteln Sie die Parameter Ihres neuen Programms (analog zu b).

2. Implementieren Sie die Anweisung $e = (a + b)(c + (a + b)d)$ auf einer Stackmaschine.

   (a) Sie haben die Befehle `push` (3 Taktzyklen), `pop` (3), `add` (1), `mult` (5) zur Verfügung.

   (b) Sie haben zusätzlich den Befehl `dup` (1 Taktzyklus) zur Verfügung, der das oberste Stackelement dupliziert. Die Addition ist kommutativ ($a + b = b + a$).

   Zählen Sie für beide Fälle die Anzahl der Befehle, die Anzahl der Speicherzugriffe (mit und ohne dem Einlesen der Befehlscodes) und die Anzahl der Taktzyklen, die das Programm benötigt. Ermitteln Sie auch, wie viele Bytes im Programmablauf zwischen Speicher und CPU transferiert werden, wenn ein Operationscode 1 Byte lang ist, Speicheradressen 2 Bytes und Operanden 4 Bytes. a, b, c und d stehen von Beginn an im Speicher.

3. Zum Vergleich der Effizienz von Speicherzugriffen bei vier verschiedenen Architekturen betrachten wir:

   (a) Akku-Maschine (Ein-Adreßsmaschine).

   (b) Speicher-Maschine (3-Adreßmaschine): Alle drei Operanden befinden sich im Speicher.

   (c) Stack-Maschine (Null-Adreßmaschine): Alle Operationen laufen über den Stack ab. Nur PUSH und POP greifen auf den Speicher zu, alle anderen Befehle holen ihre Operanden vom Stack und legen die Ergebnisse wieder dort ab.

   (d) Load-Store-Maschine (Register-Maschine): Alle Operationen finden in Registern statt, Register-Register-Befehle haben 3 Operanden. Es gibt 16 allgemeine Register, die Registerkennung ist 4 Bit lang.

   Zum Vergleich der Speicherzugriffe treffen wir für alle Architekturen folgende Annahmen:

   - Der Operationscode ist immer 1 Byte lang.
   - Alle Speicheradressen haben eine Länge von 2 Bytes.
   - Alle Operanden sind 4 Bytes lang.
   - Die Länge aller Instruktionen ist ein Vielfaches von 1 Byte.
   - Die Variablen A, B, und C stehen vom Beginn an im Speicher.
   - Mit einem Speicherzugriff können maximal 4 Bytes übertragen werden.

   Schreiben Sie für jede der 4 obigen Architekturen Assembler–Code für die Anweisung `A = B + C` . Verwenden Sie dazu die (syntaktisch der entsprechenden Architektur angepaßten) Assemblerbefehle `load, store, add, push, pop, move` und ermitteln Sie die Anzahl der Befehlsbytes und die Anzahl der Datenbytes für jede Code-Sequenz. Welche Architektur hat den effizientesten Code und welche die wenigsten Speicherzugriffe (= Befehle und Daten)?