Hochschule Konstanz Technik, Wirtschaft und Gestaltung (HTWG)
Fakultät Informatik
Rechnerarchitektur
Prof. Dr.-Ing. Michael Blaich

Übung zur Vorlesung Rechnerarchitektur AIN

Assemblerprogrammierung 1

Die Abgabe erfolgt durch Hochladen der Lösung in Moodle. Zusätzlich wird die Lösung in der Übung nach dem Abgabetermin stichprobenartig kontrolliert.

Bearbeitung in Zweier-Teams

Team-Mitglied 1: Tobias Latt
Team-Mitglied 2: Jannis Liebscher

1 Daten in Arrays

Implementieren Sie in MARS ein Assemblerprogramm, das

1. im Speicher eine Zahl *N* und ein Array *A* mit *N* Integer-Werten anlegt und
2. die Summe der *n* Werte im Array bestimmt und in das Register *$v0* schreibt

Wählen Sie als Beispiel *n=6*.

Verwenden Sie zum Anlegen der Werte im Speicher die Assembler Direktive *.word*. Laden Sie die Variable *N* sowie die Adresse des Arrays mit den Pseudo-Instruktionen *lw Register,Label* bzw. *la Register,Label*. (Kapitel 2.6 bzw. Hilfe in MARS)

2 Erste Prozedur

Implementieren Sie eine Prozedur (Label: *ISODD*) mit einem Integer-Wert *x* als Argument. Die Prozedur soll den Wert *1* zurückliefern, falls *x* ungerade ist und den Wert *0* zurückliefern, falls *x* gerade ist.
Tipp: Verwenden Sie die Instruktion *andi.*

Implementieren Sie eine zweite Prozedur (Label: *ISEVEN*), die komplementär zur ersten Prozedur arbeitet. Die Prozedur soll den Wert *1* zurückliefern, falls *x* gerade ist und den Wert *0* zurückliefern, falls *x* ungerade ist. Implementieren Sie diese Funktion, indem Sie die Funktion ISODD aufrufen und das Ergebnis invertieren.

Testen Sie die beiden Funktionen in MARS, in dem Sie sie nacheinander aufrufen und das Ergebnis in die Register $s1 und $s2 speichern.

Achten Sie auf die MIPS-Konventionen zur Implementierung von Prozeduren.

3 Prozeduren und Arrays

1. Erweitern Sie den Code aus Aufgabe 1 und legen Sie ein zweites Array *B* an, das zu Beginn mit *n* Nullen gefüllt ist.
2. Implementieren Sie eine Prozedur, die alle geraden Elemente eines Arrays *A* in ein Array *B* schreibt und die Anzahl der geraden Elemente zurückliefert. Argumente der Prozedur sind die Adressen der Arrays *A* und *B* sowie die Anzahl *n* der Elemente im Array. Der C-Code der Funktion ist unten gegeben. Halten Sie sich bei der Implementierung des Assembler-Code strikt an die Vorgaben der C-Funktion sowie die MIPS Konventionen.

```
int evenElem(int A[], int B[], int n) {

        int i=0;
        int j=0;
        while (i<n) {
                if (isEven(A[i])) {
                        B[j]=A[i];
                        j++;
                i++;
        }
        return j;
}
```

3. Testen Sie ihre Prozedur mit dem Array A=[3, 4, 6, 8, 11, 13].

4 Verständnisfragen

a) Warum kann PC-Software vorkompiliert, also als Binärpaket ausgeliefert werden und muss nicht für jede CPU, auf der sie eingesetzt wird, neu übersetzt werden?

b) Fragenkomplex Register:

i) Erklären Sie kurz, was ein Register ist. Speicherplätze als Teil der CPU in der unmittelbaren Nähe des Rechenwerks.

ii) Wie viele Register gibt es in MIPS? 32

iii) Würde Sie erwarten, dass die Anzahl der Register in Zukunft stark (entsprechend Moore's Law) ansteigt? Begründen Sie ihre Antwort.

Kein starker Anstieg, da unübersichtlich, außerdem kein Mehrwert

iv) In MIPS können nur Register als Operanden für Rechenoperationen der ALU genutzt werden. Es kann nicht direkt mit Variablen gerechnet werden, die im Hauptspeicher stehen. Nennen Sie Gründe für diese Einschränkung.

Zugriffszeit
Alu kann nicht auf Hauptspeicher zugreifen

c) Erklären Sie die folgenden Eigenschaften des MIPS Befehlssatzes:

i) Es gibt einen Befehl lbu (load byte unsigned) aber keinen Befehl sbu (store byte unsigned). ein register kann nur als ganzes adressiert werden, in einem wort haben die einzelnen bytes eigene adressen

ii) Es gibt einen Befehl sra (shift right arithmetic) aber keinen Befehl sla (shift left arithmetic).

bei negativen und positiven werten wird bei linksshift mit nullen aufgefüllt

iii) Es gibt einen Befehl addi (add immediate) aber keinen Befehl subi (subtract immediate).

subi kann durch addi mit negativer konstante umgesetzt werden