Grundlagen zur Programmieren in Assembler

TechGI 2 - SoSe 2014

Registerinitialisierung

Definition von Werten in Variablen durch:
- Variablendeklaration
  (z.B. `int i, int j = 0`)
- Wertezuweisung
  (z.B. `i = a+b`)

```
1.  int doit(int a, int b) {
2.        int i;
3.        i = 0;
4.
5.        int j = 0;
6.
7.        i = a+b;
8.
9.        j = 2*i;
10.
11.       int k=0;
12.
13.       k = 2+i;
14.
15.       return j;
    }
```

Definition von Werten in Registern durch:
- Initialisierung: Addition mit Null-Register und 0
  (z.B. `addi $t0, $0, 0`)
- Addition von zwei Werten aus Registern
  (z.B. `add  $t0, $a0, $a1`)

```
1.  doit:
2.        addi $t0, $0, 0
3.
4.
5.        add  $v0, $0, $0
6.
7.        add  $t0, $a0, $a1
8.
9.        add  $v0, $t0, $t0
10.
11.       addi $t1, $0, 0
12.
13.       addi $t1, $t0, 2
14.
15.       jr $ra
```

```
      # int a <-> $a0
      # int b <-> $a1
      # int j <-> $v0
      # int i <-> $t0
      # int k <-> $t1
```

Tipp: in Kommentaren (#) dokumentieren, welches Register welcher Variable Entspricht

Sprünge

Nur wenn eine Bedingung erfüllt ist, werden die nächsten Zeilen ausgeführt

```
1.  int doit2 (int a) {
2.        if(a>0)
3.              return 1;
4.
5.  //-------------if beendet
6.        return 0;
7.
    }
```

wenn (a>0), dann `return 1.`
ansonsten `return 0;`

Erst wenn eine Bedingung nicht erfüllt ist, werden die nächsten Zeilen übersprungen

```
1.  doit2:
2.  if:   ble   $a0, $zero, e_if
3.        addi  $v0, $zero, 1
4.        jr $ra
5.  #-----------------if beendet
6.  e_if: addi  $v0, $zero, 0
7.        jr $ra
```

wenn (NICHT(a>0)); also (a<=0)), dann überspringe

```
addi  $v0, $zero, 1
jr $ra
```

Nächste Zeile ist

```
addi  $v0, $zero, 0
jr $ra
```

Unbedingte Sprünge		
j	Jump	der Sprung wird immer ausgeführt
	Ziel	Zeile, die durch das <Label> gekennzeichnet ist
	Verwendung	Wiederholung von Schleifen
jal	Jump and Link	der Sprung wird immer ausgeführt, gleichzeitig wird die nächste Zeile als Rücksprungadresse ins `$ra` geschreiben
	Ziel	Zeile, die durch das <Label> gekennzeichnet ist
	Verwendung	Aufruf von Unterprogrammen, Rekursion
jr	Jump Register	Der Sprung wird immer ausgeführt
	Ziel	Adresse aus einem Register
	Verwendung	Rückkehr in Ober-und Hauptprogramme

Bedingte Sprünge		
beq	=	der Sprung wird ausgeführt, wenn das 1. Register gleich dem 2. Register ist
bne	≠	der Sprung wird ausgeführt, wenn das 1. Register ungleich dem 2. Register ist
blt	<	der Sprung wird ausgeführt, wenn das 1. Register kleiner dem 2. Register ist
ble	≤	der Sprung wird ausgeführt, wenn das 1. Register kleiner oder gleich dem 2. Register ist
bgt	>	der Sprung wird ausgeführt, wenn das 1. Register größer dem 2. Register ist
bge	≥	der Sprung wird ausgeführt, wenn das 1. Register größer oder gleich dem 2. Register ist