Assemblerprogrammierung

Mittwoch, 12. Juni 2019 18:28

Aufgabe 1: Kontrollstrukturen

Beschreiben Sie in Textform, durch welche Befehlsfolgen die abgebildeten Konstrukte dargestellt werden können, und geben Sie Sie anschließend entsprechenden MIPS-Assembler Befehle an:

a).

```
int i = 12; int j=0;
while(i > 0){
        j += --i;
}
```

b)

```
int a; int b; int c;
if(a ==b){
        c = a*b;
} else {
        c = a - b;
}
```

Werte von „a“ und „b“

Stehen in $a0 und $a1

c)

```
int sum = 0;
for(int i = 1; i<10; i++){
        sum += i + i/2;
}
```

a)

Schritt 1: Setze $t0 auf 12
Schritt 2: Setze $v0 auf 0 [entspricht j]
Schritt 3: Prüfe ob $t0 > 0 Falls ja weiter ab Schritt 4, sonst springe ans Ende
Schritt 4: Ziehe von $t0 "1" ab
Schritt 5: Addiere $t0 zu $v0
Schritt 6: Springe zu Schritt 3

```
.text
addi $t0, $zero, 12
addi $v0, $zero, 0
loop:
    slt $t2 , $zero, $t0
beq $t2, $zero, end
addi $t0, $t0, -1
add $v0, $t0, $v0
beq $zero, $zero, loop
end: ...<weiterer Code oder nop>...
```

b)

Schritt 1: Falls $a0 == $a1: Springe nach Schritt 4
Schritt 2: Subtrahiere $a1 von $a0, Erg. nach $v0
Schritt 3: Springe ans Ende
Schritt 4: multipliziere $a0 mit $a1, speichere Erg. nach $v0

```
.text
beq $a0, $a1, if
sub $v0, $a0, $a1
beq $zero, $zero, end
if: mult $a0, $a1
mflo $v0
end: ....<Code>....
```

c)

Schritt 1:Lege SchleifenStartVariable "1" nach $t0
Schritt 2: Lege SchleifenEndVariable "10" nach $t1
Schritt 3: Setze $v0 = "0"
Schritt 4: Falls $t0 == $t1, springe ans Ende
Schritt 5: Addiere $t0 zu $v0
Schritt 6: dividiere $t0 durch "2", Erg. nach $t2
Schritt 7: Addiere $t2 zu $v0
Schritt 8: Inkrementiere $t0 um 1
Schritt 9: Springe zu Schritt 4

```
.text
addi $t0, $zero, 1
addi $t1, $zero, 10
add $v0, $zero $zero
loop:
    beq $t0, $t1, end
add $v0, $v0, $t0
srl $t2, $t0, 1
add $v0, $v0, $t2
addi $t0, $t0, 1
beq $zero, $zero, loop
end: ...<Code>...
```

Aufgabe 2: Kontrollstrukturen in der Anwendung

Schreiben Sie ein Programm in Assemblersprache, welches über die Zahlen von 1 bis 100 iteriert. Dabei soll die Summe aller geraden Zahlen in $v0, die Summe aller ungeraden Zahlen in $v1 abgelegt werden.

```
.text
addi $t0, $t0, 1
addi $t1, $t1, 101
add $v0, $zero, $zero
add $v1, $zero, $zero

loop:
   beq $t0, $t1, end
sll $t2, $t0, 31
beq $t2, $zero, even
add $v1, $v1, $t0
beq $zero, $zero, end
even:
   add $v0, $v0, $t0
end:
   addi $t0, $t0, 1
bne $t0, $t1, loop
```

Aufgabe 3: Load / Store

An der Adresse 0x10000400 liege eine Liste mit Ganzzahlen ohne „0“. Das Ende der Liste sein durch ein Wort nur aus „Nullen“ gekennzeichnet.

Erstellen Sie ein Programm in Assemblersprache, welches die Summe aller Listenelemente bildet. Legen Sie das Ergebnis in $v0 ab.

```
.text
add $v0, $zero, $zero
li $t0, 0x10000400
move $t1, $t0

loop:
   lw $t2, ($t1)
beq $t2, $zero, end
add $v0, $v0, $t2
addi $t1, $t1, 4
beq $zero, $zero, loop
end: ...<Code>...
```

Aufgabe 4: Umsetzung von Arithmetik

Schreiben Sie ein Programm in Assemblersprache, welches die Primfaktoren zu einer in $a0 liegenden Zahl berechnet. Speichern Sie die Primfaktoren an der Adresse 0x10000000 als Liste, wobei das erste Listenelement die Anzahl der Listenelemente (ohne sich selbst) angeben soll. Es sei anzunehmen, dass an der angegebenen Adresse genug Speicherplatz reserviert wurde.

```
.text
li $t0, 0x10000000
move $t1, $t0
addi $t1, $t1, 4
addi $t2, $zero, 2
sw $zero, ($t1)
move $t8, $a0
addi $t7, $zero, 1
loop:
    beq $t8, $t7, end
div $t8, $t2
mfhi $t3
beq $t3, $zero, factor
addi $t2, $t2, 1
beq $zero, $zero, loop
factor:
    lw $t4, ($t0)
addi $t4, $t4, 1
sw $t4, ($t0)
sw $t2, ($t1)
addi $t1, $t1, 4
mflo $t8
beq $zero, $zero, loop
end: ...<Code>....
```