Maschinenprogrammierung

Mittwoch, 12. Juni 2019 18:28

Aufgabe 1: Maschinenbefehle lesen

Stellen Sie zunächst die angegebenen Maschinenbefehle in Binärform dar, und beschreiben Sie anschließend deren Funktion textuell:

- 0C000020
- 01095025
- 000841C2
- 00001010

0C000020 = 000011 00000000000000000000100000 Format I: 6-bit Opcode, 26-bit Wert/Addresse

Beschreibung: Springe zur in Wert codierten Adresse und speichere aktuellen Programmschrittzähler in $ra ($31)
Mnemoic: jal 0x00000020

01095025 = 000000 01000 01001 01010 00000 100101 Format R: 6-bit Opcode, 3x5-bit Register, 5-bit Shamt, 6-bit Function

Beschreibung: Vergleiche die Inhalte von Register $t0 ($8) und $t1 ($9) mit OR, und speichere das Ergebnis in $t2 ($10)
Mnemonic: or $t2, $t0, $t1

000841C2 = 000000 00000 01000 01000 00111 000010 Format R

Beschreibung: Verschiebe den Inhalt von $t0 "logical" um 7 nach rechts und speichere das Ergebnis in $t0
Mnemonic: srl $t0, $t0, 7

00001010 = 000000 00000 00000 00010 00000 010000 Format R

Beschreibung: Kopiere den Inhalt des 'hi' Registers nach $v0 ($2)
Mnemonic: mfhi $v0

Aufgabe 2: Maschinenbefehle erkennen

Geben Sie die Maschinenbefehle an, die das beschriebene Verhalten bewirken (in Hexadezimaldarstellung):

- Addiere Register 5 mit der Zahl 27 und lege das Ergebnis in Register 3
- Vergleiche Register 8 und Register 15 mit XOR und lege das Ergebnis in Register 2
- Wenn Register 9 und Register 10 gleichen Inhalt besitzen, überspringe den nächsten Befehl
- Speichere die Zahl 0x12345678 in Register 2 (2 Befehle nötig, keine Pseudobefehle!)

Addiere Register 5 mit 27 und lege Ergebnis in Register 3:

--> Format I, Immediatebefehl

--> Opcode rs rt Wert

--> Mnemonic = 'addi' -->Opcode = 001000

==> 001000 00011 00101 0000000000011001

==> 20650019

Vergleiche Register 8 und Register 15 per XOR, Ergebnis nach Register 2

--> Format R, Registerbefehl

--> Opcode rs rt rd shamt Function

-->Mnemonic = 'xor' --> Opcode = 000000, Funtion = 100110

==> 000000 01000 01111 00010 00000 100110

==> 010F1026

Wenn Register 9 und Register 10 den gleichen Inhalt besitzen (Inhalt 9 == Inhalt 10), überspringe den nächsten Befehl

--> Format I, Immediatebefehl

-->Opcode rs rt Wert

Mnemonic: 'beq' --> Opcode = 000100

==> 000100 01001 01010 0000000000000001

==> 112A0001

Speichere die Zahl0x12345678 in Register 8 (ohne Pseudobefehle

--> Format I, Immediatebefehl

--> Opcode rs rt Wert

Mnemonic: lui --> Opcode = 001111 ; addi --> Opcode = 001000

==> 001111 00000 01000 0001001000110100
==> 001000 01000 01000 0101011001111000

Aufgabe 3: Maschinenprogramme

In Register $a0 liegt eine unbekannte gültige Zahl x.

Schreiben Sie Maschinenprogramme, die folgende Funktionen leisten. Wenn die Bedingung erfüllt ist lege ‘1‘ in $v0, sonst ‘0‘ in $v0.

- Prüfe ob x negativ ist.
- Prüfe ob x restlos durch 2 teilbar ist.
- Prüfe ob x eine “gültige“ Sprungadresse darstellt.

a) Passendes Mnemonic: "slt" (set less than)
--> Format R, Function = 101010

Befehl 1: 000000 00100 00000 00010 00000 101010

b) Mehrere Mögllichkeiten: Division->Restbetrachtung; Bit-Logische Betrachtung

Bit-Logische Betrachtung ist effizienter und "schöner"
Mnemonic: "andi"
--> Format I

Befehl 1: 001100 00100 01000 0000000000000001 Vergleiche $a0 per AND mit "1", Erg. nach $t0

Jetzt liegt in $t0 entweder "0" wenn die Zahl gerade ist, oder "1" wenn die Zahl ungerade ist

Mnemonic: "xori"
Befehl 2: 001110 01000 00010 0000000000000001 Vergleiche $t0 per XOR mit "1", Erg. nach $v0

Wenn $t0 = "1", dann ergibt XOR mit "1" -> "0"; Wenn $t0 = "0" ergibt XOR mit "1" -> "1"

c) Wieder mehrere Möglichkeiten, im Prinzip ähnlich zu Aufgabe b) : Division-> Restbetrachtung; Bit-Logische Betrachtung (per Vergleich oder per Shift)

Hier zur Demonstration per Shift-Verfahren

Gültige Adresse definiert sich als jede Adresse die mit "..00" endet; da Wortlänge = 32 Bit; Adressierung über Byte --> Zwischen Adresse 0400 und 0404 liegen 4 Byte -> 32 Bit

--> Prüfung auf die niedrigsten 2 bit

Mnemonic "sll" (shift left logical)

Befehl 1: 000000 00000 00100 01000 11110 000000 Schiebe Inhalt von $a0 um 30 Stellen nach links, Erg. nach $t0

Jetzt ist Register $t0 entweder "0", falls Wert in $a0 eine gültige Adresse ist, oder "nicht null" falls nicht

Befehl 2: 000101 00000 01000 0000000000000010 Falls $t0 != $zero : überspringe nächsten 2 Befehle

Befehl 3: 001000 00000 00010 0000000000000001 Addiere zu $zero den Wert "1" und speichere Erg. nach $v0

Befehl 4: 000100 00000 00000 0000000000000001 Überspringe den nächsten Befehl

Befehl 5: 001000 00000 00010 0000000000000000 Addiere zu $zero den Wert "0" und speichere Erg. nach $v0

Befehle 2 -5 könnten in Hochsprache etwa so dargestellt werden:

```
if($t0 != $zero)      Befehl 2
{
      $v0 = 0;        Befehl 5
} else
{
      $v0 = 1;        Befehl 3
}
```

Aufgabe 4:

Erstellen Sie ein Maschinenprogramm, welches folgende Formel ausdrückt:

$$\sum_{k=1}^{100} (-1)^k * k$$

Das Ergebnis soll schlussendlich in Register $v0 vorliegen.

a) Beschreiben Sie zunächst in Textform, welche Schritte notwendig sind und Welche Maschinenbefehle in Frage kommen. Legen Sie bereits die verwendeten Register fest.

b) Notieren Sie das Machinenprogramm in Hexadezimaldarstellung.

a)

Programmablauf:
Schleife von 1 bis 100, bei jedem Durchlauf soll der aktuelle Wert abwechselnd auf- addiert oder -subtrahiert werden. Das Ergebnis soll in $v0 stehen

Vorbereitung:
Schritt 1: Lege Startwert der Schleife "1" nach $t0 [entspricht Laufvariable k]
Schritt 2: Lege Endwert der Schleife "101" nach $t1 [da Schleife 100tste Iteration noch durchführen soll
Schritt 3: Lege positiv/negativ Zähler mit Startwert "-1" nach $t2 [$-1^k$ entspricht abwechselndem +/-]
Schritt 4: Setzte $v0 auf "0"

Schleife:
Schritt 5: Berechne $t0 * $t2
Schritt 6: Kopiere aus "lo" nach $t3
Schritt 7: addiere $t3 zu $v0
Schritt 8: erhöhe $t0 um "1"
Schritt 9: Setze $t2 = "+1" falls $t0%2 == 0; sonst $t2 = "-1"
    9.1 $t0 AND "1" -> $t4
    9.2 wenn $t4 == $zero, springe zu "Schritt 9.5"
    9.3 setze $t2 = "-1"
    9.4 Springe zu "Schritt 10"
    9.5 setze $t2 = "+1"
Schritt 10: Falls $t0 != $t1, springe zu Schritt 5 zurück

b)

Hexcode:	MIPS-R2000:	[alternativ mit Labels:]
20080001	addi $t0, $zero, 1	
20090065	addi $t1, $zero, 101	
200AFFF	addi $t2, $zero, -1	
20020000	addi $v0, $zero, 0	
010A0018	mult $t0, $t2	loop: mult $t0, $t2
00005812	mflo $t3	
004B1020	add $v0, $v0, $t3	
21080001	addi $t0, $t0, 1	
310C0001	andi $t4, $t0, 1	
11800002	beq $t4, $zero, 2	beq $t4, $zero, if
200AFFF	addi $t2, $zero, -1	
10000001	beq $zero, $zero, 1	beq $zero, $zero, ifend
200A0001	addi $t2, $zero, 1	if: addi $t2, $zero, 1
1509FFF6	bne $t0, $t1, -10	ifend: bne $t0, $t1, loop