HOCHSCHULE KONSTANZ TECHNIK, WIRTSCHAFT UND GESTALTUNG (HTWG)
Fakultät Informatik
Rechner- und Kommunikationsnetze
Prof. Dr.-Ing. Michael Blaich

Übung zur Vorlesung Rechnerarchitektur AIN

Theorieübung – Single Cycle CPU

Bearbeitung in Zweier-Teams

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

Aufgabe 1: Dekodierung und Ausführung einer Instruktion

In Abbildung 1 sind Datenpfad und Steuerwerk einer einfachen Eintakt-Implementierung der MIPS CPU dargestellt. Die Abbildung zeigt ebenfalls den Inhalt des Instruktionsspeichers. Der Inhalt der Register ist in

0	0x0	4	0x8	8	0x15	12	0x6	16	0x2	20	0x0	24	0x0	28	0x100
1	0x3	5	0x2	9	0x17	13	0x0	17	0x3	21	0x0	25	0x0	28	0x800
2	0x4	6	0x8	10	0x0	14	0x200	18	0x80	22	0x0	26	0x0	30	0x200
3	0xA	7	0x0	11	0x0	15	0x0	19	0x0	23	0x0	27	0xFF	31	0x10

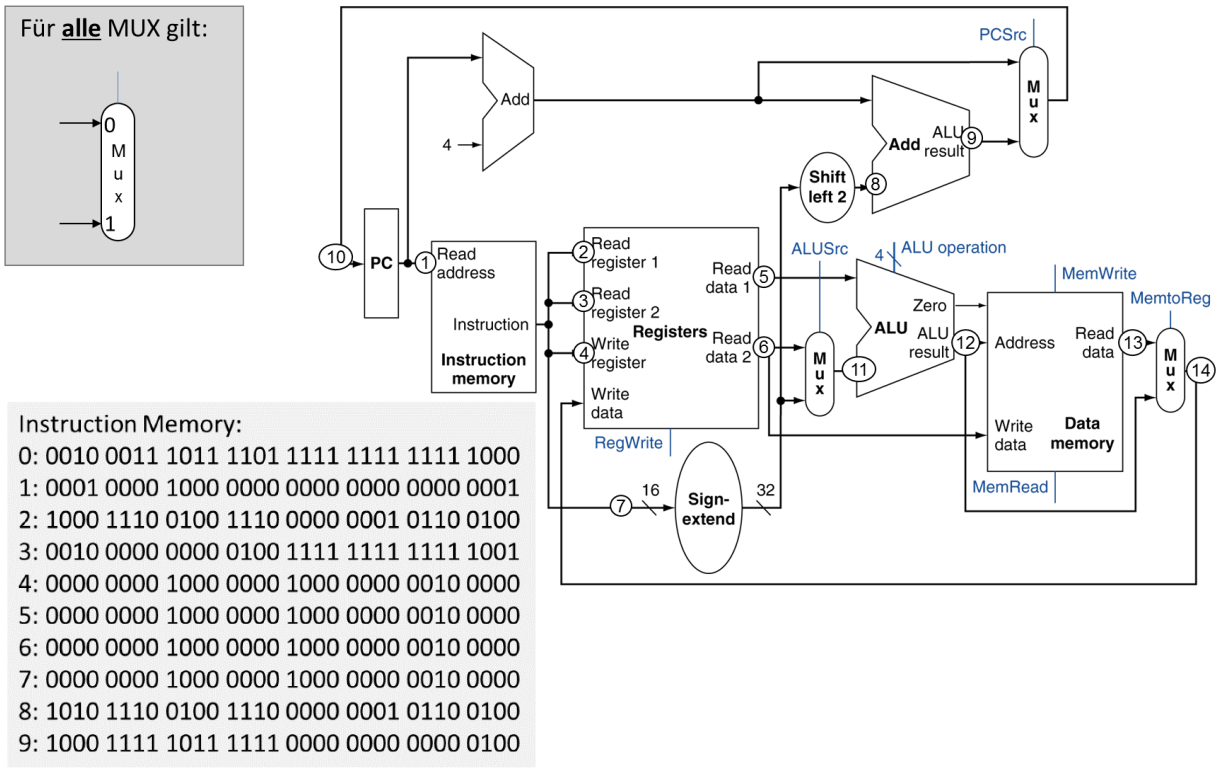
Tabelle 1 gegeben.

1) Bestimmen Sie die Instruktion, die im nächsten Takt ausgeführt wird, wenn im Register $pc der Wert 8 steht und geben Sie den zugehörigen Befehl in Assembler an.

```
100001110.01001110.00000001.01100100    lw $s2, 356($t6)
```

2) In Abbildung 1 sind einige Signale des Datenpfads mit Nummern in Kreisen versehen. Tragen Sie in der linken Hälfte von Tabelle 2 die Werte ein, die diese Signale bei der Ausführung der Instruktion im nächsten Takt annehmen.

3) Tragen Sie in der rechten Hälfte von Tabelle 2 die Werte ein, die die Steuersignale bei der Ausführung der Instruktion im nächsten Takt annehmen. Geben Sie eine kurze, stichwortartige Erklärung an, welche Unterscheidungen mit den jeweiligen Steuersignalen getroffen werden.



Instruction Memory:
0: 0010 0011 1011 1101 1111 1111 1111 1000
1: 0001 0000 1000 0000 0000 0000 0000 0001
2: 1000 1110 0100 1110 0000 0001 0110 0100
3: 0010 0000 0000 0100 1111 1111 1111 1001
4: 0000 0000 1000 0000 1000 0000 0010 0000
5: 0000 0000 1000 0000 1000 0000 0010 0000
6: 0000 0000 1000 0000 1000 0000 0010 0000
7: 0000 0000 1000 0000 1000 0000 0010 0000
8: 1010 1110 0100 1110 0000 0001 0110 0100
9: 1000 1111 1011 1111 0000 0000 0000 0100

Abbildung 1

0	0x0	4	0x8	8	0x15	12	0x6	16	0x2	20	0x0	24	0x0	28	0x100
1	0x3	5	0x2	9	0x17	13	0x0	17	0x3	21	0x0	25	0x0	28	0x800
2	0x4	6	0x8	10	0x0	14	0x200	18	0x80	22	0x0	26	0x0	30	0x200
3	0xA	7	0x0	11	0x0	15	0x0	19	0x0	23	0x0	27	0xFF	31	0x10

Tabelle 1: Registerinhalte in Hexadezimaldarstellung

Signale des Datenpfads (nummeriert)				Steuersignale	
1	2/8	8	1424	ALUSrc	1
2	14	9	1436	ALU operation	0010
3	18	10	3/12	PCSrc	0
4	18	11	356	MemWrite	0
5	128	12	484	MemtoReg	0
6	512	13	Mem[484-487]	MemRead	1
7	356	14	Mem[484-487]	RegWrite	1

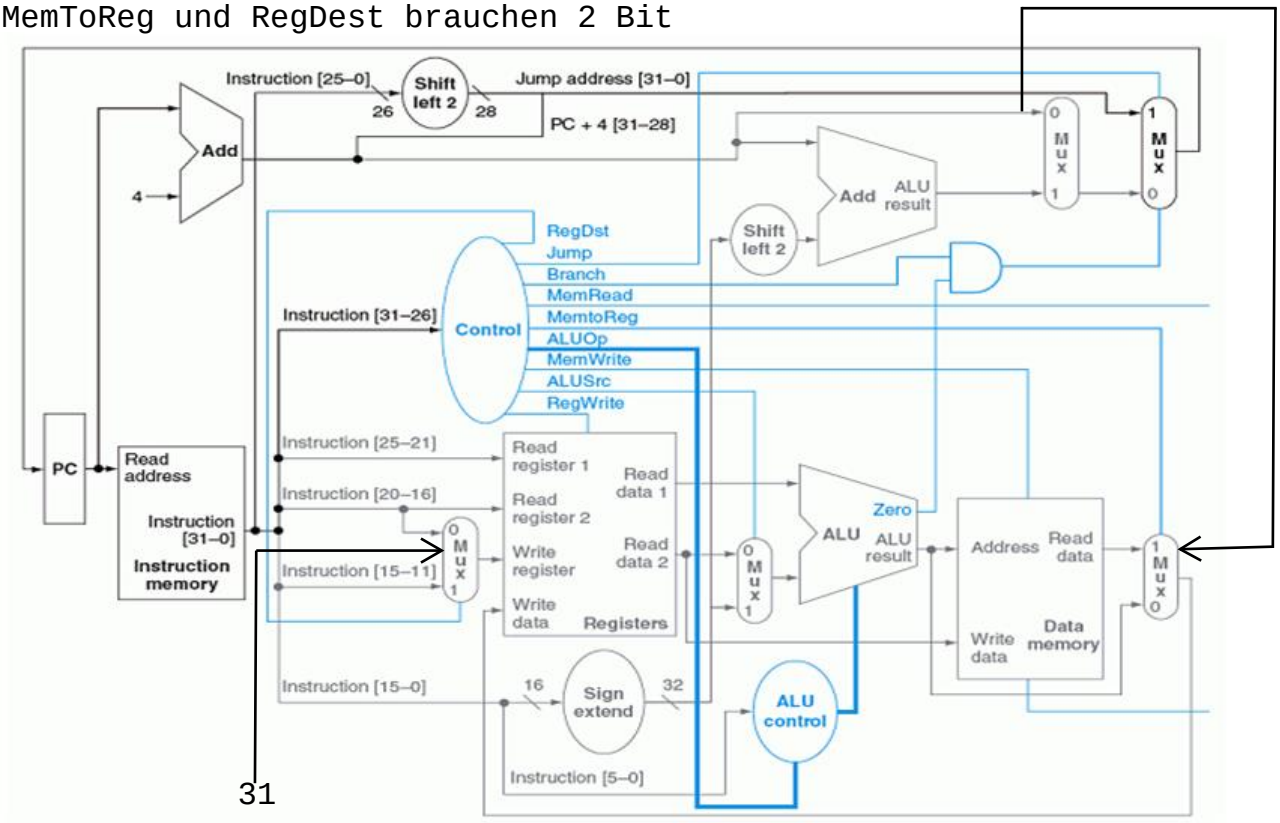
Tabelle 2: Ausführung der Instruktion

Aufgabe 2: Jump and Link

Mit der Instruktion *jal LABEL* wird eine Prozedur aufgerufen. Format und Funktionsweise der jal Instruktion sind ähnlich zur jump(j) Instruktion. Allerdings ist der OPCODE 3 und die Instruktion speichert zusätzlich zum Sprung die Rücksprungadresse, also die Adresse der nächsten Instruktion (PC+4), im Register $ra. Erläutern Sie anhand einer Skizze, welche Erweiterungen Sie in Datapath und Control der idealisierten MIPS CPU vornehmen müssten, um die Instruktion jal zu realisieren.

Description:	Jumps to the calculated address and stores the return address in $31
Operation:	$31 = PC + 4; PC = (PC & 0xf0000000) | (target << 2)
Syntax:	jal target
Encoding:	`0000 11ii iiii iiii iiii iiii iiii iiii`

`MemToReg und RegDest brauchen 2 Bit`



Aufgabe 3: Verständnisfragen

1. Welche der folgenden Aussagen trifft auf einen Ladebefehl zu?

    a. MemtoReg muss so gesetzt werden, dass Daten aus dem Speicher an den Registersatz gesendet werden.

    b. MemtoReg muss so gesetzt werden, dass das richtige Registerziel an den Registersatz gesendet wird.

    c. Wir kümmern uns nicht um das Setzen von MemtoReg.

2. Wahr oder falsch?

    a. Main Control ist ein kombinatorisches Element mit 6 Eingängen (dem Function-Code) und acht Ausgängen, wobei wie gesehen ALUOp aus zwei Bits besteht.

    b. Bei einem „add“ Befehl gibt die ALU-Control eine „0010“ aus

    c. Bei einem add Befehl wird im Datenblock wird das Steuersignal "MemWrite" auf "1" gesetzt.

    d. Auf den PC werden prinzipiell 4 Worte addiert, um die nächste Instruktion aus dem Instruktionsspeicher zu laden.

3. Betrachten Sie den folgenden Befehl:
   Befehl: AND Rd, Rs, Rt
   Interpretation: Reg[Rd] = Reg[Rs] AND Reg[Rt]

    a. Was sind die Werte des Steuersignals, das für den obigen Befehl von der Steuerung (RA KAP 3 - Folie 6) generiert wird?

    b. Welche Ressourcen (Blöcke) führen für diesen Befehl eine nützliche Funktion aus? `Alle außer Data memory`

    c. Welche Ressourcen (Blöcke) erzeugen Ausgaben, die nicht für diesen Befehl verwendet werden? Welche Ressourcen erzeugen für diesen Befehl keine Ausgaben? `Data memory`

```
zu a:
RegWrite:1,ALUSSrc:0,MemWrite:0,ALUOp:00,MemtoReg:0,
MemRead:0,Branch:0,Jump:0,RegDst:1
```

4. Die einfache Eintakt-MIPS-Implementierung in (RA KAP 3 - Folie 6) kann nur manche Befehle implementieren. Neue Befehle können zu einer existierenden Befehlssatzarchitektur hinzugefügt werden, aber die Entscheidung, dies zu tun oder zu lassen, hängt unter anderem von den Kosten und der Komplexität ab, die dem Datenpfad und dem Steuerwerk des Prozessor hierdurch aufgeladen werden. Die drei Teilaufgaben dieser Aufgabe beziehen sich auf die folgenden neuen Befehle:
   Befehl: LWI Rt, Rd(Rs)
   Interpretation: Reg[Rt] = Mem[Reg[Rd]+Reg[Rs]]

   a. Welche existierenden Blöcke (wenn es denn solche gibt) können für diesen Befehl verwendet werden? `Instruction memory, Registers, ALU, Data memory`
   b. Welche neuen funktionalen Blöcke (wenn es denn solche gibt) benötigen wir für diesen Befehl? `Multiplexer für RD nach Read Register 2`
   c. Welche neuen Signale vom Steuerwerk (wenn es denn solche gibt) benötigen wir für die Unterstützung dieses Befehls? `Steuersignal für neuen Multiplexer`