SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Jan Reineke
Dr. Andreas Abel
Lucas Biehl

Systemarchitektur SS 2021

Präsenzblatt 7 (Lösungsvorschläge)

Hinweis: **Dieses Aufgabenblatt wurde von Tutoren erstellt. Die Aufgaben sind für die Klausur weder relevant noch irrelevant, die Lösungsvorschlage weder korrekt noch inkorrekt.**

Aufgabe 7.1: MIPS Kontrollpfad 1

Im Folgenden betrachten wir R-type MIPS Instruktionen, sowie die Instruktionen `lw`, `sw`, `beq` und `j`.

1. Was charakterisiert R-type-Instruktionen?
2. Welchen Einfluss hat es auf diese Instruktionen, wenn
   - `MemtoReg` konstant auf 0 steht?
   - `RegWrite` konstant auf 1 steht?
   - `AluSrc` konstant auf 0 steht?

Lösungsvorschlag:

1. Alle R-Typ-Instruktionen haben die Form `op rd, rs, rt`. Alle Eingabedaten kommen aus Registern (`rs` und `rt`). Mit der ALU wird ein neuer Wert berechnet und dann im Zielregister (`rd`) gespeichert.
2. - Alle Instruktionen außer `lw` funktionieren noch. Die Instruktion `lw` funktioniert nicht mehr, da nichts mehr vom Speicher in ein Register geschrieben werden kann.
   - Nun wird bei jeder Instruktion ein Register beschrieben. Allerdings sollte das nicht immer passieren. Somit wird es bei `sw`, `beq` und `j` zu falschem Verhalten kommen.
   - `AluSrc` wird von `lw` und `sw` zur Berechnung der Adresse verwendet. Wenn dies nicht mehr funktioniert, werden diese zwei Operationen nicht mehr funktionieren.

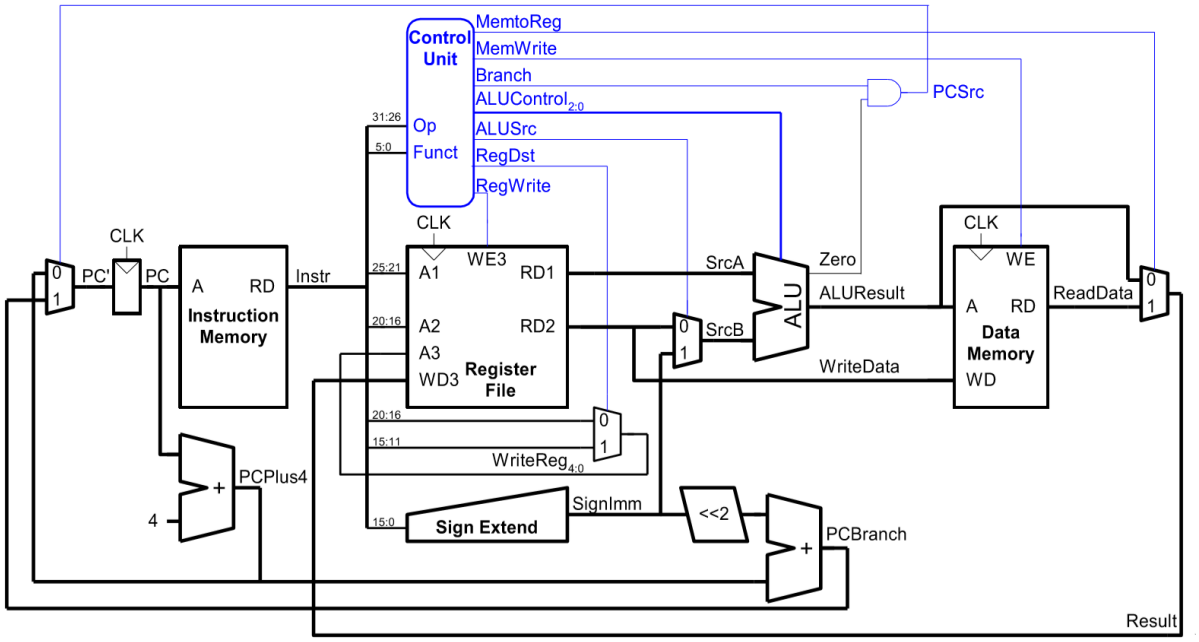
Aufgabe 7.2: MIPS Kontrollpfad 2

Wie überprüft die Instruktion `beq` (wie in der Vorlesung) auf Gleichheit? Wie könnte man es so ändern, dass Gleichheit durch `xor` überprüft wird?

Lösungsvorschlag: Man muss sich hierzu den `AluOp` code anschauen. Dieser wird auf 01 gesetzt, was eine Subtraktion bedeutet. Um dies zu einem `xor` zu ändern, könnte man die unused bits 11 mit `xor` belegen. Die Kontrollbits wären dann wie bei `beq`, außer dass `AluOp` auf 11 gesetzt wird.

Aufgabe 7.3: MIPS Kontrollpfad 3

Betrachten Sie den folgenden Einzeltakt-Prozessor, der in der Vorlesung behandelt wurde.



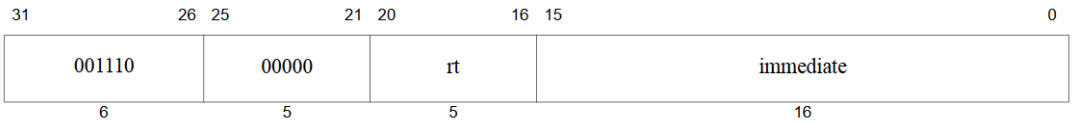
Wir führen in dieser Aufgabe eine eigene Tabelle für ALUOp und ALUControl ein.

ALUOp	Funct	ALUControl	Bedeutung
00	X	001	add
01	X	010	shift: SrcA « 16
10	X	011	shift: SrcB « 16
01	00001	110	slt
11	X	110	SrcA + 1

Hinweis: Diese Tabelle wurde eigens für diese Aufgabe erstellt. Sie sollte nicht für die Bearbeitung des Projektes oder der Klausur verwendet werden. Generell sollte diese Aufgabe nicht als Grundstein für Entscheidungen im Projekt verwendet werden!

Füllen Sie die untere Tabelle für die folgenden beiden Befehle aus:

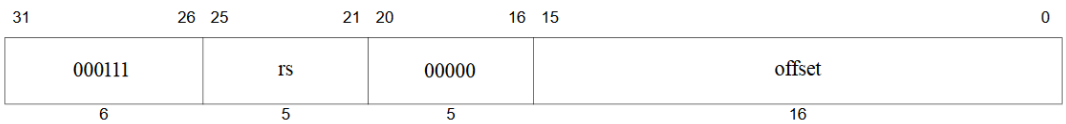
1. `svui` – store value in upper immediate



   Der Wert des Registers `$rt` soll in die Speicherzelle mit der Adresse `immediate « 16` gespeichert werden.

   Beispiel: Im Register $3 ist der Wert 14 gespeichert. Der Befehl 001110 00000 00011 00000000 01011100 speichert den Wert 14 in die Speicherzelle $92 \cdot 2^{16} = 6029312$.

2. `beqz` – branch equal zero



Es soll genau dann abgezweigt werden, wenn der Wert im Register `$rs` $= 0$ ist.

Instruktion	Op $_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUOP$_{1:0}$
`svui`								
`beqz`								

Lösungsvorschlag:

Instruktion	Op $_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUOP$_{1:0}$
`svui`	001110	0	X	1	0	1	X	10
`beqz`	000111	0	X	0	1	0	X	00

Bei `beqz` macht man sich zunutze, dass die Bits 16 bis 20 immer auf 0 gesetzt sind. Dadurch wird beim Addieren von A und B für B der Inhalt des Registers 0 verwendet. Dieses ist nach Definition immer 0. Für A wird der Inhalt des Registers `$rs` verwendet. Daher ist das Ergebnis von $A + 0$ genau dann 0, wenn $A = 0$ ist.

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Jan Reineke
Dr. Andreas Abel
Lucas Biehl

System Architecture SS 2021

Tutorial Sheet 7 (Suggested Solutions)

Note: **This task sheet was created by tutors. The tasks are neither relevant nor irrelevant for the exam, the suggested solutions are neither correct nor incorrect.**

Problem 7.1: MIPS Control 1

In the following, we consider R-type MIPS instructions, as well as the instructions `lw`, `sw`, `beq`, and `j`.

1. What characterizes R-type instructions?
2. What influence does it have on these instructions if
   - `MemtoReg` is constantly set to 0?
   - `RegWrite` is constantly set to 1?
   - `AluSrc` is constantly set to 0?

Suggested solution:

1. All R-type instructions are of the form `op rd, rs, rt`. All input data comes from registers (`rs` and `rt`). The ALU is used to calculate a new value and then store it in the destination register (`rd`).
2. - All instructions except `lw` still work. The instruction `lw` does not work anymore because nothing can be written from memory to a register.
   - Now a register is written with every instruction. However, this should not always happen. Thus wrong behavior will occur with `sw`, `beq` and `j`.
   - `AluSrc` is used by `lw` and `sw` to calculate the address. If this stops working, these two operations will stop working.

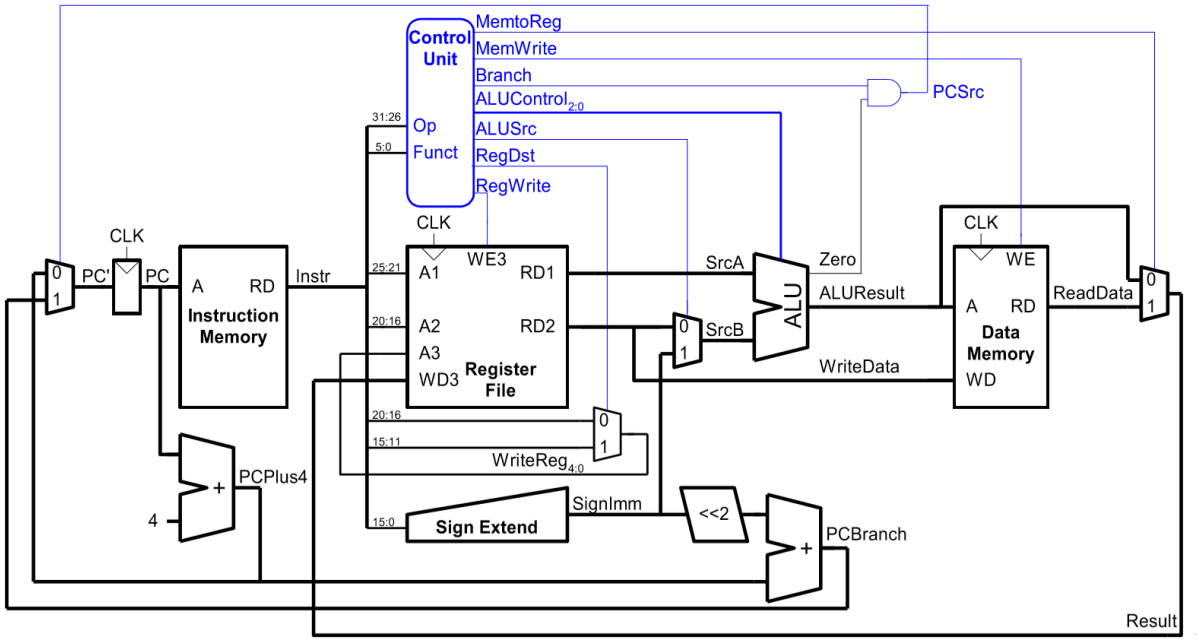
Problem 7.2: MIPS Control 2

How does the instruction `beq` check for equality (as in the lecture)? How could it be changed so that equality is checked by `xor`?

Suggested solution: You have to look at the `AluOp` code for this. This is set to 01 which means subtraction. To change this to a `xor`, one could assign the unused bits 11 to `xor`. The control bits would then be as for `beq` except that `AluOp` is set to 11.

Problem 7.3: MIPS Control 3

Consider the following single-cycle processor that has been discussed in the lecture.



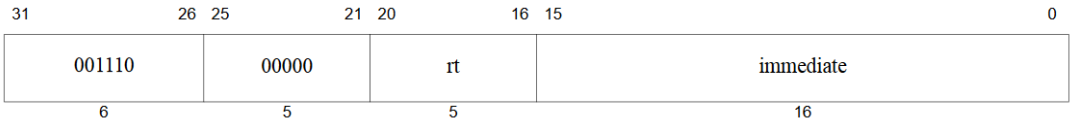
In this task we introduce a separate table for ALUOp and ALUControl.

ALUOp	Funct	ALUControl	Meaning
00	X	001	add
01	X	010	shift: SrcA « 16
10	X	011	shift: SrcB « 16
01	00001	110	slt
11	X	110	SrcA + 1

Note: This table was created specifically for this assignment. It should not be used for working on the project or the exam. In general, this task should not be used as a cornerstone for decisions in the project!

Fill in the table below for the following two commands:

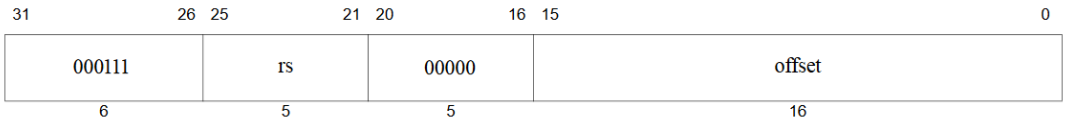
1. `svui` – store value in upper immediate



   The value of register `$rt` is to be stored in the memory cell with address `immediate « 16`.

   Example: In register $3 the value 14 is stored. The command 001110 00000 00011 000000 01011100 stores the value 14 into the memory cell $92 \cdot 2^{16} = 6029312$.

2. `beqz` – branch equal zero



   It should be branched exactly if the value in register `$rs` is 0.

Instruction	Op $_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUOP$_{1:0}$
`svui`								
`beqz`								

Suggested solution:

Instruction	Op $_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUOP$_{1:0}$
`svui`	001110	0	X	1	0	1	X	10
`beqz`	000111	0	X	0	1	0	X	00

With `beqz` one uses that the bits 16 to 20 are always set to 0. This means that the content of register 0 (always 0) is used when adding B. For A the content of register `$rs` is used. Therefore, the result of $A + B$ is 0 exactly if $A = 0$.