Übung 7

Abgabe bis **Donnerstag, 2. Juni 10:00** via EPIIC: `http://ep.iic.jku.at`.

SystemVerilog Dateien müssen vor der Abgabe in EPIIC ausgeführt werden. Dateien, die nicht kompilieren können nicht abgegeben werden.

1. ALU Implementierung (4 Punkte)

Die in der Vorlesung vorgestellte ALU (siehe Abbildung 1) muss leicht angepasst werden, damit sie in der MIPS Architektur verwendet werden kann:

- Das Eingangscarry *c* soll aus dem *select* Signal berechnet werden (0 bei Addition und 1 bei Subtraktion) und kein eigener Eingang sein.
- Ein zusätzlicher Ausgang *zero* wird benötigt, der 1 ist wenn das Resultat der Berechnung 0 ist.
- Das MSB (most significant bit) des Ausgangs wird nicht benötig.

Vervollständige das Modul *ALU* in der Datei *building_blocks.sv* um die modifizierte ALU in SystemVerilog zu implementieren. Mit Ausnahme des enthaltenen Addierers ist eine strukturelle Beschreibung des Moduls erforderlich.

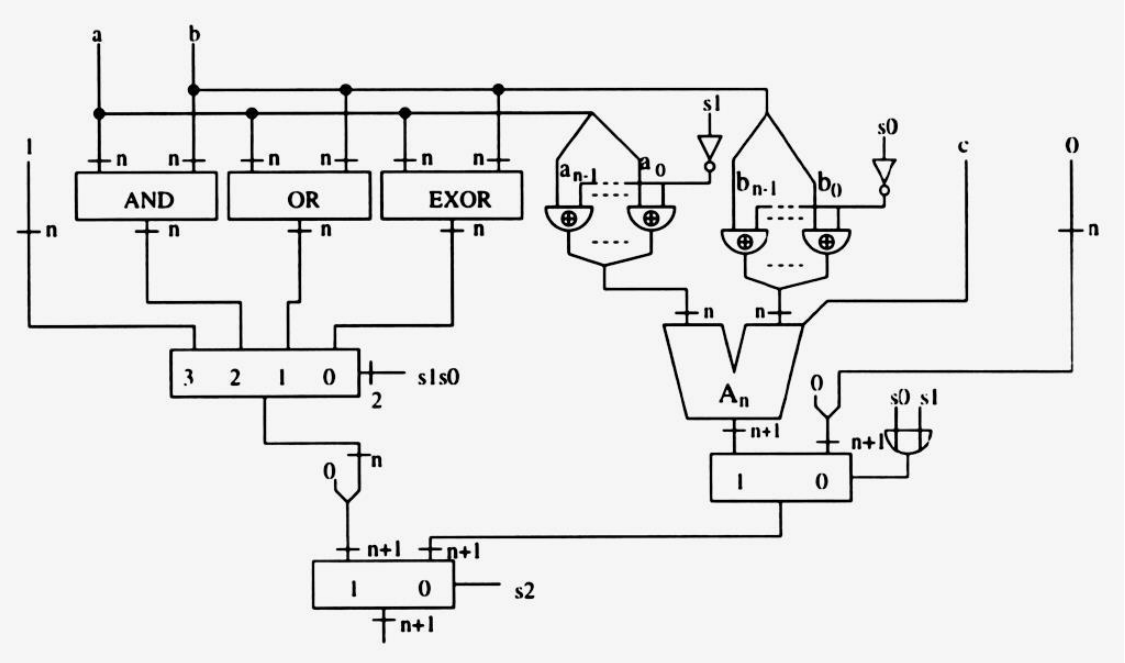

Abbildung 1: Arithmetic Logic Unit (ALU)

2. (12 Punkte)

Ziel dieser Aufgabe ist es den MIPS Prozesor aus Abbildung 2 in SystemVerilog zu implementieren. Implementierungen der Komponenten des Datenpfades (mit Ausnahme der ALU) sind in der Datei *building_blocks* vorhanden und sollen im Modul *mips* (in der Datei *mips.sv*) zusammengefügt werden. Dazu ist auch das Modul *control_unit* zu vervollständigen, das die Signale zur Steuerung des Datenpfades generiert. Folgende Instruktionen müssen unterstützt werden:

Tabelle 1: Unterstützte Instruktionen

(a) MIPS Instruktionen

OP-code	Name	Beschreibung
000000	R-Typ	R-Typ Instruktionen
000010	j	jump
000100	beq	Branch if equal
001000	addi	add immediate
100011	lw	load word
101011	sw	store word

(b) R-Typ Instruktionen

Funktionscode	Name
100000	add
100010	sub
100100	and
100101	or
100110	xor

Verwende das Modul *testbench* aus der Datei *test_sum.sv* um die Implementierung zu testen. Das Modul führt das Programm *sum.s*, das die Summe der ungeraden Zahlen in einem Array addiert, auf dem MIPS Prozessor aus. Bei der Simulation wird auch die Datei *sum.dat* benötigt. Diese Datei enthält das Array und dessen Länge.

3. Erweiterung des MIPS Prozessors (8 Punkte)

Erweitere den MIPS Prozessor in Abbildung 2 um die Befehle `jal` und `jr` um Funktionsaufrufe zu ermöglichen. Die beiden Befehle haben folgende Semantik:

- `jal` ist eine J-Typ Instruktion (`OP-code = 000011`) die `PC+4` in Register `$ra` (31) schreibt und `PC` auf `{(PC+4)[31:28], target, 2'b0}` (identisch zum Befehl `j`) setzt.
- `jr` ist eine R-Typ Instruktion (`funct = 001000`) mit der Semantik `PC = [rs]`.

Füge die Erweiterungen auch der Implementierung aus Aufgabe 2 hinzu und teste den erweiterten Prozessor mit dem Modul *testbench* aus der Datei *test_ackermann*. Das Modul führt das Programm *ackermann.s*, das die Ackermann Funktion für die Parameter 3 und 2 berechnet (siehe Übung 2), aus und überprüft das Resultat.

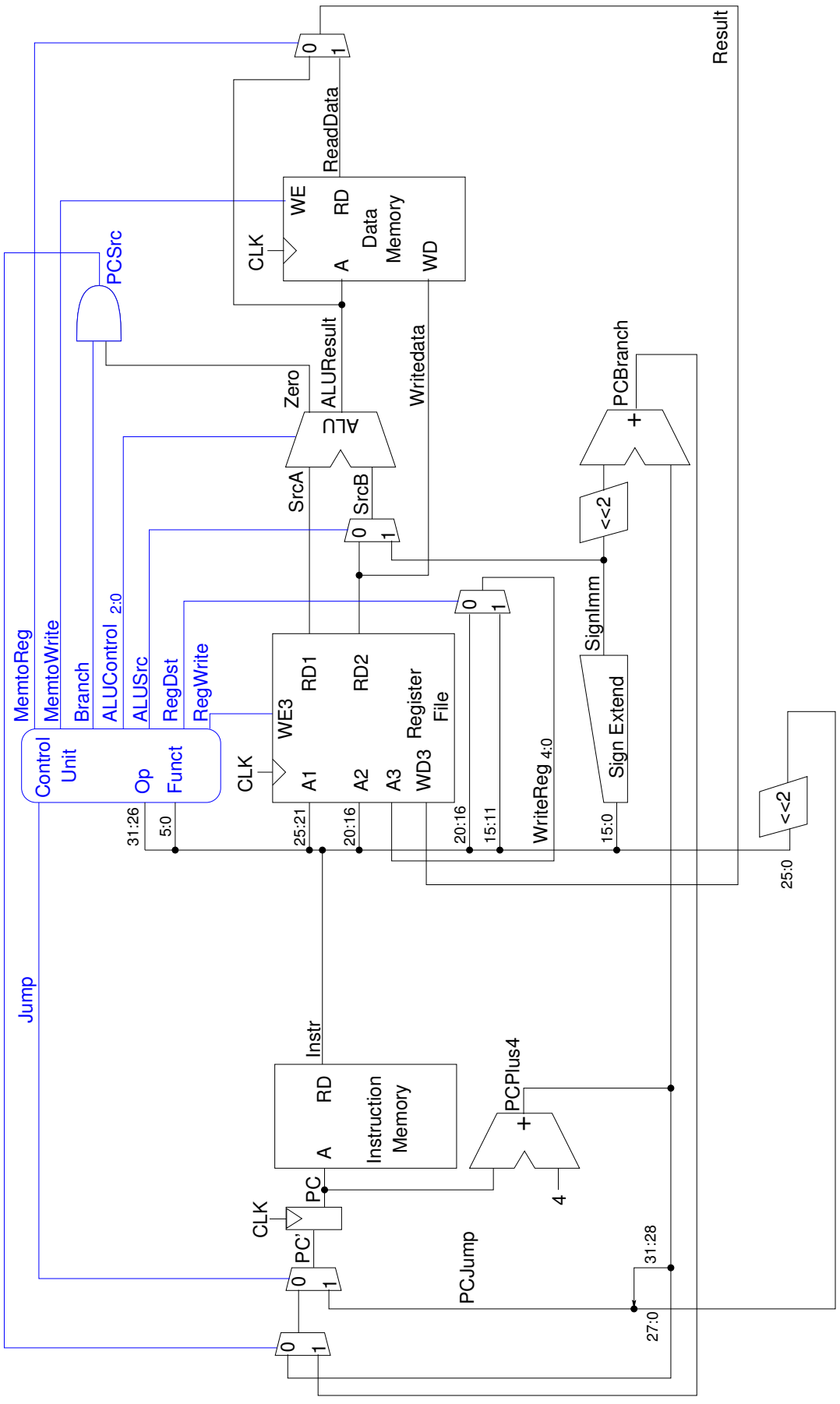

Abbildung 2: MIPS Prozessor