## Projekt

Abgabe bis Donnerstag, 20. Juni 10:15 via EPIIC: https://ep.iic.jku.at.

## Realisierung eines MIPS-Prozessors (48 Punkte)

Im Rahmen des Projekts soll der in der Vorlesung vorgestellte MIPS-Prozessor erweitert und in VHDL realisiert werden, sodass die Funktion *titlecase* aus Übung 2 ausgeführt werden kann. Das Projekt ist in die folgenden drei Teile unterteilt:

# 1. Übersetzung von Assembler Code in Maschinensprache

Ubersetze die Datei *titlecase.s* mit Hilfe von QtSpim in Maschinensprache und speichere diese in der Datei *imem.dat* ab. Übersetze zusätzlich die globalen Daten in Binärformat (kann ebenfalls mit Hilfe von QtSpim gemacht werden) und speichere diese in der Datei *dmem.dat* ab. In den beiden Dateien beschreiben die ersten 8 Zeichen jeder Zeile ein Datenwort (hexadezimal codiert). Alle weiteren Zeichen in einer Zeile werden ignoriert und können z.B.: für Kommentare genutzt werden. In beiden Dateien ist das zu verwendende Format exemplarisch angeführt. Folgendes ist bei dieser Aufgabe zu beachten:

- Der Speicherbereich für den Programmcode startet bei der Adresse 0x0, d.h. die erste Instruktion der main-Methode soll sich an der Adresse 0x0 befinden. Deshalb müssen die Sprungadressen von J-Typ Befehlen entsprechend angepasst werden.
- Der Speicherbereich für globale Daten startet an der Adresse 0x100. Der String *str* soll daher an dieser Adresse liegen. Deshalb muss der Pseudobefehl 1a mit addi (anstelle von 1ui) ersetzt weden.
- QtSpim weicht bei der Befehlscodierung von bedingten Sprüngen (z.B.: beq) vom MIPS Standard ab. Deshalb muss hier bei dem von QtSpim generierten Maschinenwort die 16-Bit Konstante um 1 verringert werden.

### 2. Konzeptionelle Erweiterung des Prozessors

Der in der Vorlesung vorgestellte MIPS-Prozessor muss erweitert werden, um den die Funktion *titlecase* aus Übung 2 ausführen zu können.

Erweitere den in der Vorlesung vorgestellten MIPS-Prozessor (Abbilding 2), sodass auch die Befehle jal, jr, slti, bne, lb und sb unterstützt werden. Erweitere dazu in Abbildung 2 für jeden dieser Befehle (falls nötig) den Datenpfad und füge weitere Kontrollsignale hinzu. Gib für jeden Befehl auch den Wert aller Kontrollsignale an. Nimm dabei an, das die ALU aus Abbildung 1 verwendet wird und gib don't cares explizit mit x an.

<sup>&</sup>lt;sup>1</sup>Der in schwarz gezeichnete Teil in Abbildung 2 visualisiert den Datenpfad, während der Kontrollpfad in blau gezeichnet ist.

Die Abgabe beinhaltet ein Bild mit dem erweitertem Daten- und Kontrollpfad des Prozessors pro zusätzlichem Befehl (es können auch mehrere Erweiterungen in ein Bild gezeichnet werden) sowie eine Tabelle mit den Werten aller Kontrollsignale für die neuen Befehle.

Hinweis: Entnimm die Semantik der einzelnen MIPS Befehle der in EPIIC zur Verfügung gestellten Befehls-Referenz. Ändere für die Realisierung von 1b und sb die Schnittstelle des Datenspeichers nicht. Es wird weiterhin eine 32-Bit Adresse (muss ein vielfaches von 4 sein) am Eingang A angelegt und das entsprechende Datenwort am Ausgang RD ausgegeben. Falls ein Datenwort geschrieben werden soll (bei der nächsten positiven Taktflanke), so muss es am Eingang WD angelegt und das Signal WE auf 1 gesetzt werden.

#### 3. Realisierung in VHDL und Test

Implementiere den erweiterten MIPS-Prozessor in VHDL. Verwende dazu die im Template vorgegebenen Strukturen und füge (wenn nötig) weitere Komponenten hinzu. Zu vervollständigende Abschnitte im VHDL-Code sind mit TODO gekennzeichnet. Durch das erweitern des Prozessors kann es sein, dass sich die Schnittstelle der Komponenten datapath und controller (verwendet in der Architecture der Komponente mips) ändern. Teste den erweiterten Prozessor, indem du das in Aufgabe 1 übersetze Programm ausführst. Eine entsprechende Testbench ist bereits im Template verfügbar.

Hinweis: Damit der VHDL Code richtig kompiliert, muss der VHDL Standard in Modelsim auf 1076-2008 geändert werden. Bei der Simulation in EPIIC müssen auch die .dat Dateien ausgewählt werden.

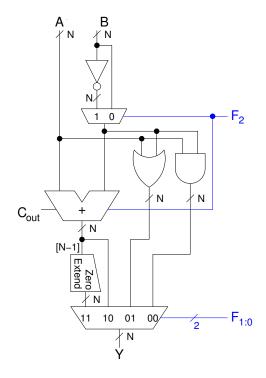


Abbildung 1: ALU

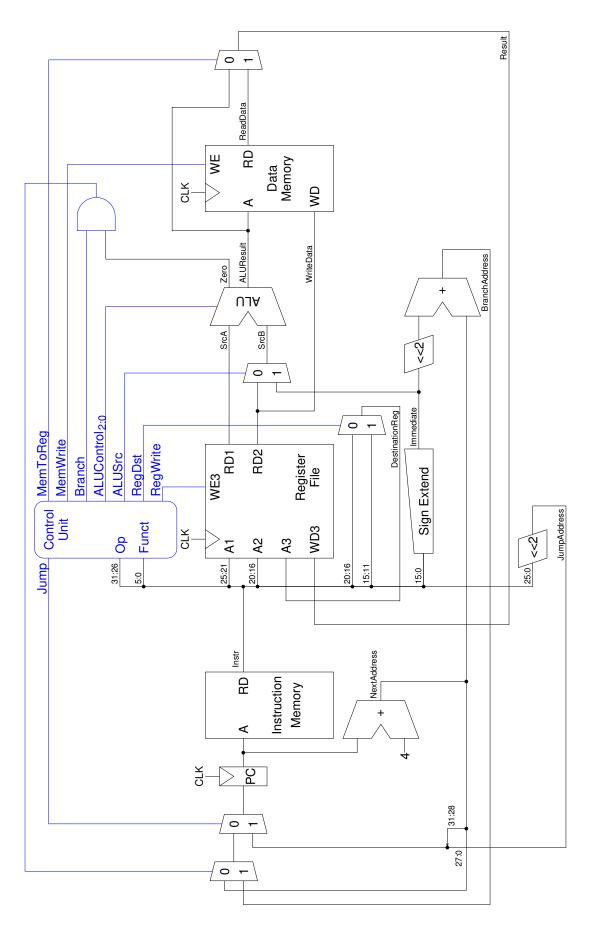


Abbildung 2: MIPS Prozessor