10. Blatt	Fachgebiet Architektur eingebetteter Systeme **Rechnerorganisation Praktikum**	AES Embedded Systems Architecture WS2023/24

Ausgabe: 22. Januar 2024

Abgaben

Theorie	28. Januar 2024
Praxis	11. Februar 2024
Rücksprache	12./13. Februar 2024

Auf diesem Aufgabenblatt können insgesamt **20 Punkte** erreicht werden. Daher beträgt die Bearbeitungszeit der Praxisaufgaben **zwei Wochen**. Die Rücksprache für das gesamte Blatt findet in dem Termin nach der Abgabe des gesamten Blattes statt.

Aufgabe 1: Mehrzyklenprozessoren (5 Punkte)

Mehrzyklenarchiketuren wurden entwickelt, um die Leistung von Einzyklusarchitekturen zu verbessern. Informieren Sie sich (z.B. in [1, ab Seite 273]) über Leistungsbegrenzungen von Singlecycle-Architekturen und der Entwicklung von Multicycle-Implementierungen. Beantworten Sie die folgenden Fragen:

1. (3 Punkte) Timingperformance ist der wichtigste Grund, warum Einzyklus-Prozessoren heutzutage höchst selten zu finden sind. Erklären Sie das hauptsächlichen Leistungsproblem einer Singlecycleimplementierung. Geben Sie ein numerisches Beispiel an, das erlaubt, dieses Problem abzuschätzen.
2. (1 Punkt) Wie verbessert eine Mehrzyklenimplementierung dieses Problemen?
3. (1 Punkt) Ist Timingperformance die einzige Verbesserung einer Mehrzyklenimplementierung? Wie verhält es sich mit den benötigten Hardwareressourcen und der Leistungsaufnahme?

Mehrzyklenarchiketuren sind etwas anderes als *Pipelining*. In dieser Aufgabe geht es **nicht** um Pipelining. Beide Ansätze dienen der Verbesserung der Timingperformance, unterscheiden sich jedoch in der Umsetzung, in der konkreten Performanceverbesserung und in ihren Vor- und Nachteilen.

Aufgabe 2: ALU Control Testbench (5 Punkte)

In dieser Aufgabe sollen Sie nun einmal selbst eine Testbench entwerfen. Als Vorgabe erhalten Sie die zu testende Komponente `aluCtrlExt`. Dabei handelt es sich um die bereits von Ihnen implementierte `aluCtrl`, jedoch erweitert, um weitere Instruktionen zu implementieren. In Tabelle 1 erhalten Sie die zugehörige Wahrheitstablelle der optimierten Schaltung. Da eine Testbench grundsätzlich über keine Ports verfügt, werden diese hier nicht spezifiziert.

Implementieren Sie ihre Testbench in der Datei `aluCtrlExt_tb.vhd` und führen Sie die Testbench durch einen Aufruf von `make clean all` aus:

1. (1 Punkt) Instanziierung und Verdrahtung des zu testenden Moduls

aluOp (1:0)	f (5:0)	operation (3:0)	jr	Beschreibung
00	------	0010	0	lw/sw
01	------	0110	0	bew
10	--000-	0010	0	add(u)/nop
10	--001-	0110	0	sub(u)
10	--0100	0000	0	and
10	---101	0001	0	or
10	--011-	1100	0	nor
10	--1000	0001	1	jr
10	--1001	0001	0	
10	--101-	0111	0	slt(u)
10	--1100	0001	0	
10	--111-	1101	0	
11	------	----	-	

Tabelle 1: Wahrheitstabelle aluCtrlExt

2. (1 Punkt) Anlegen eines Arrays von (mind. 16) Testvektoren und Durchlaufen des Arrays in einer Schleife
3. (1 Punkt) Erkennen von Fehlern durch Vergleich mit einem Erwartungswert
4. (1 Punkt) Ausgabe einer Fehlermeldung, wenn das Ergebnis vom Erwartungswert abweicht
5. (1 Punkt) Auswertung am Ende der Testbench (inkl. eindeutiger Aussage, ob das Modul korrekt funktioniert)

Hinweis: Treffen Sie keine Annahmen für das Verhalten von `aluCtrlExt` bei einem don't-care! Ihre Testcases sollen hauptsächlich das Verhalten bei gültigen und implementierten Instruktionen prüfen. Die vollständigen *function codes* liegen in der `mipsISA` als Konstanten vor und sollen auch als solche genutzt werden.

⌨ Aufgabe 3: Mehrzyklenimplementierung (10 Punkte)

Realisieren Sie den gesamten MIPS-Prozessor in der Mehrtaktimplementierung. Verbinden Sie dazu alle notwendigen Komponenten in der Architektur `structural`, wie es in Abbildung 1 dargestellt ist. Nutzen Sie die vorgegebene Datei `mipsCpu_mc.vhd`.

Hinweise:

- Es ist möglicherweise einfacher, den gesamten Datenpfad neu zu implementieren, statt die Eintaktimplementierung anzupassen.
- Nutzen Sie die in dem Paket `proc_config` definierten Konstanten, welche u.a. die Breite des PC sowie die Anzahl der Elemente im RAM definieren. Außerdem sind dort die Datentypen der Debug-Ports einzelner Module beschrieben.
- Der RAM arbeitet als gemeinsamerInstruktions- und Datenspeicher, da er sowohl das Programm- als auch das Datensegment eines MIPS-Programms enthält.

- Damit die IF-Stufe und die MEM-Stufe innerhalb eines Taktes ausgeführt werden können, muss der Instruktions- und Datenspeicher mit dem invertierten Taktsignal angesteuert werden.
- Beachten Sie, dass der Instruktionsspeicher weniger Einträge hat, als mit dem PC adressiert werden könnten. Üblicherweise wird sich der PC nur in diesem Adressbereich bewegen. Überlegen Sie sich, welcher Teil des PC-Registers zur Adressierung des Speichers verwendet werden muss. Dabei ist außerdem zu beachten, dass der PC eine Byteadresse enthält, während der Instruktionsspeicher wortweise adressiert wird.

Implementieren Sie den Datenpfad entsprechend der Abbildung 1. Verbinden Sie außerdem die Debug-Ports von `mipsCtrlFsm`, `regFile` und `ram` mit den entsprechenden Debug-Ports des MIPS, um der Testbench den Zugriff auf den Prozessorstatus zu ermöglichen. Schließen Sie außerdem die Ausgänge der `bin2Char`-Module an den Debug-Port für die 7-Segement-Anzeige an. Testen Sie ihre Implementierung mit Hilfe der vorgegebenen Testbench `mipsCpu_mc_tb`, indem Sie im Aufgabenordner das Kommando `make clean all` ausführen.

Dabei wird der Prozessor mit einem vollständigen Programm getestet, wobei mehrere zur Verfügung. Diese decken teilweise alle, **teilweise nur einige der implementierten Instruktionen ab**. Das ausgeführte Programm wird in der Testbench im Generic `init_file_name` definiert, welches Sie modifizieren müssen, um das entsprechende Programm auszuwählen. Folgende Programme stehen zur Verfügung:

- `memcpy.s.mif`: Lade- und Speicherbefehle, NOP
- `clip.s.mif`: alle Instruktionen
- `bubblesort.c.mif`: alle Instruktionen

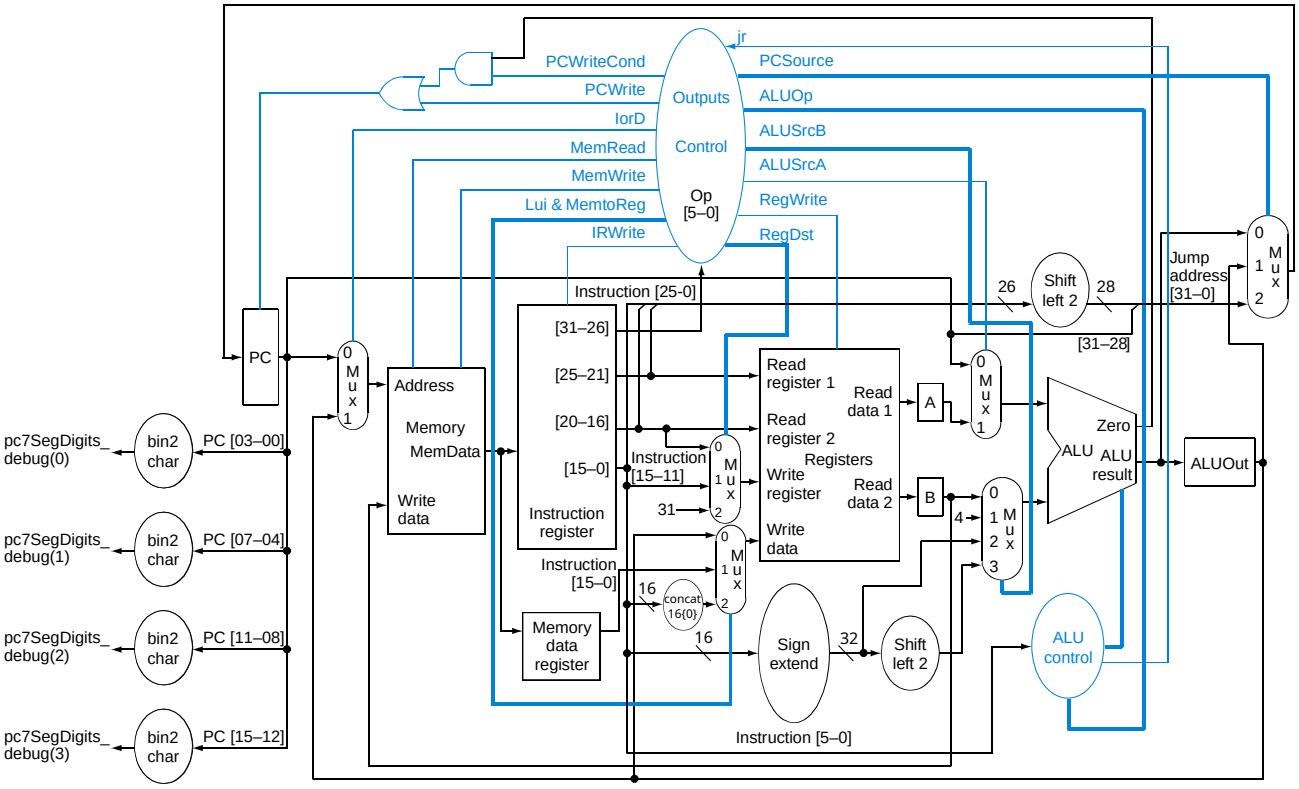

Abbildung 1: Der einfache MIPS mit Steuereinheit, nach [1, S. 266]

Literatur

[1] David A. Patterson and John L. Hennessy. *Rechnerorganisation und -entwurf.* Spektrum Akademischer Verlag, September 2005.