Fachgebiet Technische Informatik	**Praktikum Prozesssorarchitektur Wintersemester 2018/2019**	Brandenburg University of Technology Cottbus - Senftenberg

Projekt: VHDL-Implementierung eines RISC Prozessors

In diesem Praktikum soll ein einfacher RISC Prozessor entwickelt werden. Die Verarbeitungsbreite des Prozessors soll 32-Bit und das Befehlswort soll 21-Bit betragen und er soll insgesamt 25 Instruktionen beherrschen. Es gibt dabei 4 unterschiedliche Arten von Instruktionen. Dazu zählen Register- Register, Speicher/Lade- , Immediate- und Sprung/Verzweigungs- Befehle. Intern sollen diesem Prozessor 32 allgemeine Register zur Verfügung stehen. Der Prozessor soll eine Harvard-Architektur darstellen und hat aus diesem Grund zwei getrennte Speicherschnittstellen(Datenspeicher, Programmspeicher). Die gesamte Architektur soll mit einer DLX-Pipeline ausgestattet werden, um die Abarbeitungsgeschwindigkeit des Prozessors zu steigern. Dabei ist nur die Basis-Pipeline mit 5 Stufen(IF, ID, EX, MA, WB) zu realisieren. Weiterhin sollen folgende Beipässe implementiert sein:

- EX→ID
- MA→ID
- WB→ID

Ebenfalls soll die Pipeline für den Assembler-Code transparent sein. Das bedeutet, dass alle Hazards, die nicht durch die Beipässe verhindert werden, durch das automatische einfügen von *stall* Befehlen aufzulösen sind. Das zu entwerfende Prozessormodell muss synthetisierbar sein!

Aufgabe:

Es ist ein Verhaltensmodell auf der RT- Ebene zu erstellen. Mit Hilfe des beiliegenden Assemblers können verschiedene SW-Routinen implementiert werden, um die Funktionsweise des Prozessors zu validieren. Bei der Abgabe ist zu beachten, dass mindestens die Beispielroutine ordnungsgemäß abgearbeitet wird. Dies lässt sich durch einen Vergleich des Datenspeicherinhaltes, produziert von eurer Architektur, mit dem mitgelieferten Speicherabbild überprüfen.

Vorgaben:

- Befehlssatz und Kodierung
- Assembler und Beispielroutine
- Testbench und Bibliothek
- Programmspeicher *prog.mem* und Datenspeicher *data.mem* sowie Speicherabbild des Datenspeichers *data_mem_after_sim.mem* für die Beispielroutine nach der Simulation mit dem Makro *sim.do*

wichtige Daten:

- Endgültige Abgabe des Prozessorentwurfs: 13.01.2019 (als `zip` Archiv per Email)
- Abschließendes Gespräch: 15.-18.01.2019 (nach Absprache)
- Sowohl Abgabe des Entwurfs als auch das abschließende Gespräch sind Pflicht!

Instruction Set der zu entwerfenden Architektur:

1. NOP	6. XOR	11. SEQ	16. SGE	21. SUBI
2. ADD	7. MOV	12. SNE	17. LOAD	22. BEQ
3. SUB	8. SLL	13. SLT	18. STORE	23. BNE
4. AND	9. SRL	14. SLE	19. MVI	24. JUMP
5. OR	10. SRA	15. SGT	20. ADDI	25. CALL

Befehlstypen der Architektur:

1. Register-Register Instructions (R)

OP	FN	Rd	Rs1	Rs2	
2 – bit	4 – bit	5 – bit	5 – bit	5 – bit	
20	18	14	9	4	0

2. Memory-Access Instructions (M)

OP	FN	Rd	Rs	unused
2 – bit	4 – bit	5 – bit	5 – bit	5 – bit

3. Immediate Instructions (I)

OP	FN	Rd	Immediate
2 – bit	4 – bit	5 – bit	10 – bit

4. Jump/Branch Instruction (J)

OP	FN	Rd	Rs	unused
2 – bit	4 – bit	5 – bit	5 – bit	5 - bit

Instruction Encoding

Instruction	Operation	OP	FN	Type
NOP	No operation	00	0000	R
ADD Rd, Rs1, Rs2	$[Rd] \Leftarrow [Rs1] + [Rs2]$	00	0001	R
SUB Rd, Rs1, Rs2	$[Rd] \Leftarrow [Rs1] - [Rs2]$	00	0010	R
AND Rd, Rs1, Rs2	$[Rd] \Leftarrow [Rs1]$ AND $[Rs2]$	00	0011	R
OR Rd, Rs1, Rs2	$[Rd] \Leftarrow [Rs1]$ OR $[Rs2]$	00	0100	R
XOR Rd, Rs1, Rs2	$[Rd] \Leftarrow [Rs1]$ XOR $[Rs2]$	00	0101	R
MOV Rd, Rs1	$[Rd] \Leftarrow [Rs1]$	00	0110	R
SLL Rd, Rs1, Rs2	$[Rd] \Leftarrow [Rs1]$ sll $[Rs2]$	00	0111	R
SRL Rd, Rs1, Rs2	$[Rd] \Leftarrow [Rs1]$ srl $[Rs2]$	00	1000	R
SRA Rd, Rs1, Rs2	$[Rd] \Leftarrow [Rs1]$ sra $[Rs2]$	00	1001	R
SEQ Rd, Rs1, Rs2	$[Rd] \Leftarrow 1$ if $[Rs1] = [Rs2]$ else 0	00	1010	R
SNE Rd, Rs1, Rs2	$[Rd] \Leftarrow 1$ if $[Rs1] \neq [Rs2]$ else 0	00	1011	R
SLT Rd, Rs1, Rs2	$[Rd] \Leftarrow 1$ if $[Rs1] < [Rs2]$ else 0	00	1100	R
SLE Rd, Rs1, Rs2	$[Rd] \Leftarrow 1$ if $[Rs1] \leq [Rs2]$ else 0	00	1101	R
SGT Rd, Rs1, Rs2	$[Rd] \Leftarrow 1$ if $[Rs1] > [Rs2]$ else 0	00	1110	R
SGE Rd, Rs1, Rs2	$[Rd] \Leftarrow 1$ if $[Rs1] \geq [Rs2]$ else 0	00	1111	R
LOAD Rd, Rs	$[Rd] \Leftarrow mem[Rs]$	01	0000	M
STORE Rd, Rs	$Mem[Rd] \Leftarrow [Rs]$	01	0001	M
MVI Rd, #imm10	$[Rd] \Leftarrow [imm10]$	10	0000	I
ADDI Rd, #imm10	$[Rd] \Leftarrow [Rd] + [imm10]$	10	0001	I
SUBI Rd, #imm10	$[Rd] \Leftarrow [Rd] - [imm10]$	10	0010	I
BEQ Rd, Rs	$[PC] \Leftarrow [Rd]$ if $[Rs] = 0$	11	0000	J
BNE Rd, Rs	$[PC] \Leftarrow [Rd]$ if $[Rs] \neq 0$	11	0001	J
JUMP Rd	$[PC] \Leftarrow [Rd]$	11	0010	J
CALL Rd, Rs	$[Rs] \Leftarrow [PC], [PC] \Leftarrow [Rd]$	11	0011	J