Abschlusstest WS22/23

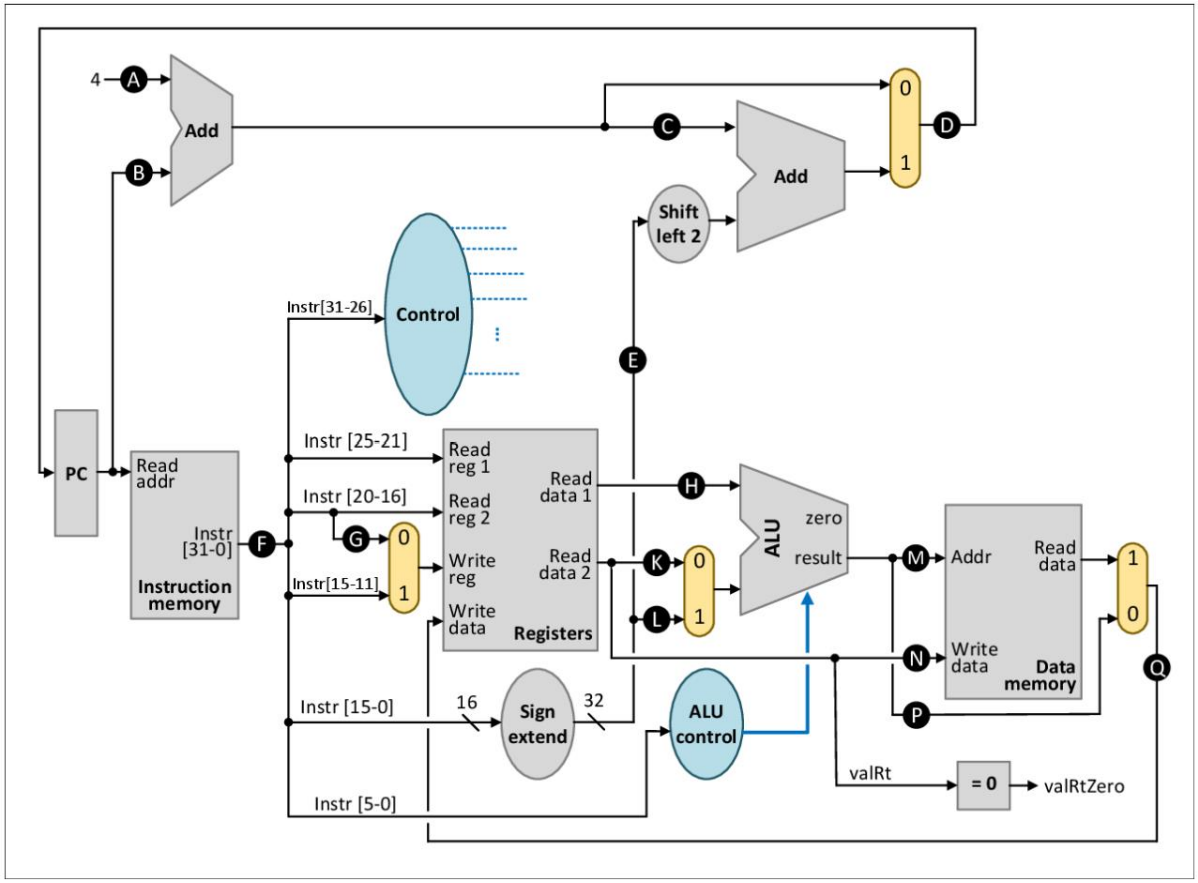
1. **Der abgebildete Assemblercode soll in Maschinencode übersetzt werden. Die Adresse der Maschinenbefehle sind angegeben. Vervollständigen Sie die Tabelle um die zwei fehlenden Werte des *imm*-Felds. Bitte tragen Sie Dezimalzahlen ohne führende Nullen ein.**

Assemblercode	Adresse	Maschinencode			
L1:					
andi $t0, $a0, 15	50	opcode = 12	rs = 4	rt = 8	imm = 15
addi $t1, $t0, -12	54	opcode = 8	rs = 8	rt = 9	imm = -12
beg $t1, $zero, L2	58	opcode = 4	rs = 0	rt = 9	imm = ___
addi $t1, $t0, -14	62	opcode = 8	rs = 8	rt = 9	imm = -14
addi $v0, $v0, 1	66	opcode = 8	rs = 2	rt = 2	imm = 1
L2:					
addi $v0, $v0, -2	70	opcode = 8	rs = 2	rt = 2	imm = -2
L3:					
addi $v0, $v0, 1	74	opcode = 8	rs = 2	rt = 2	imm = 1
srl $a0, $a0, 1	78	opcode = 0	rs = 0	rt = 4	rd = 4|shamt = 1|funct = 2
bne $a0, $zero, L1	82	opcode = 5	rs = 0	rt = 4	imm = ___

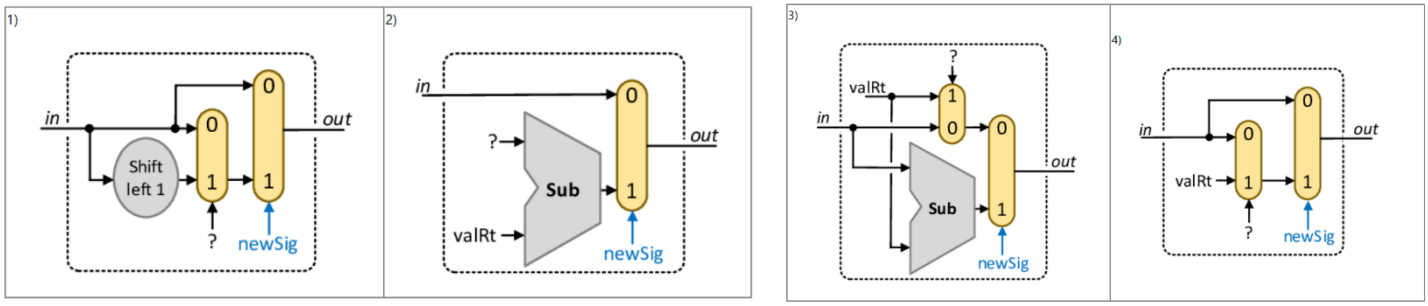
2. **Der in der linken Tabellenspalte abgebildete Assemblercode soll in Maschinencode übersetzt (assembliert) werden. Bitte vervollständigen Sie die Tabelle. Alle Werte müssen in Dezimalform eingetragen werden!**

Assemblercode	Adresse	Maschinencode					
andi $t0, $t0, 15	1000	opcode = ___	rs =___	rt = ___	imm = ___		
srl $t2, $t0, 3	1004	opcode = ___	rs = 0	rt = ___	rd =___	shamt = ___	funct = ___
ori $t0, $a2, 1	1008	opcode = ___	rs = ___	rt = ___	imm = ___		
or $t2, $t0, $t2	1012	opcode = 0	rs = 8	rt = 10	rd = 10	shamt = 0	funct = ___
sll $t1, $t2, 10	1016	opcode = 0	rs = 0	rt = ___	rd = ___	shamt = ___	funct = ___

3. **Erweitern Sie den bekannten MIPS-Eintaktprozessor, um den neuen Befehl `Add Immediate Unless Zero´ (aiuz) zu unterstützen:**
   - **aiuz soll ein Befehl im I-Format sein.**
   - **Für aiuz soll der Opcode 0x15 verwendet werden**
   - **aiuz soll sich wie der bekannte addi-Befehl verhalten, allerdings soll das Ergebnis der Addition nur dann ins Zielregister übernommen werden, falls der zu überschreibende Wert im Zielregister ungleich Null ist. Anderenfalls soll das Zielregister den vorherigen Registerwert beibehalten.**



Um die Aufgabe zu lösen, müssen Sie einen der folgenden Bausteine (1 bis 4) an eine der markierten Stellen (A bis Q) des Prozessors hinzufügen. Folgende stehen zur Verfügung:

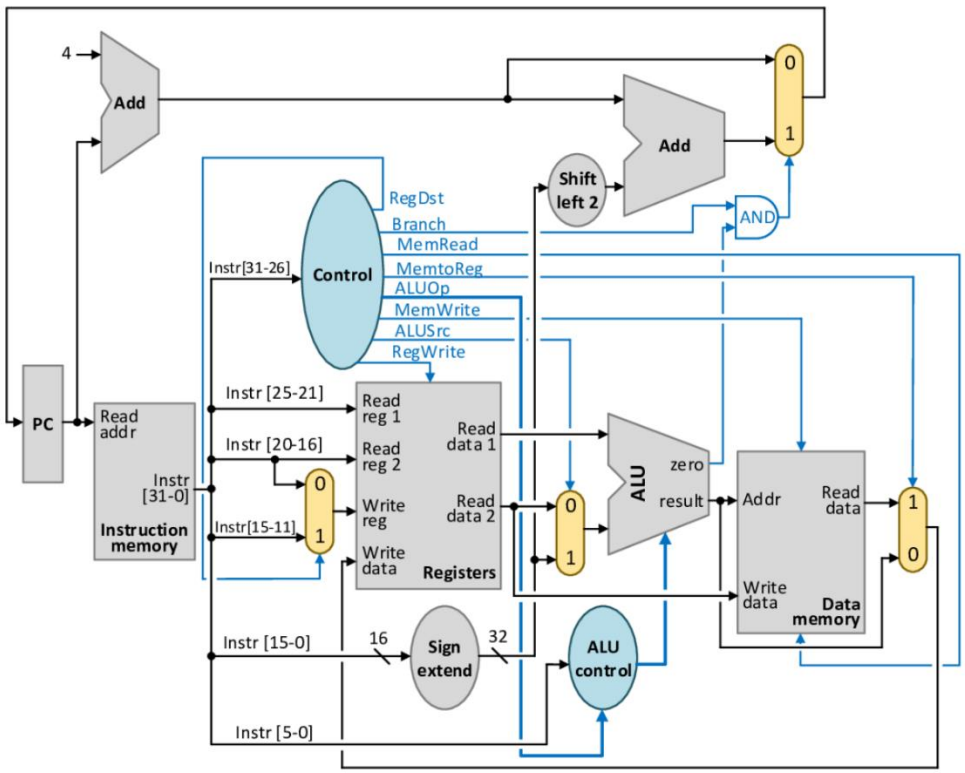


Baustein ___ soll an die Stelle ___ eingefügt werden.

Das Fragezeichen (?) im eingefügten Baustein wird mit _____ verbunden.

Mit dem neuen Steuersignal newSig soll der Prozessor zwischen addi und aiuz unterscheiden können. newSig muss von dem Block _____erzeugt werden. Die Unterscheidung zum addi-Befehl erfolgt mittels _____. Falls ein aiuz-Befehl erkannt wird, muss newSig den Wert ___ annehmen.

4.



Leider kam es bei dem abgebildeten Eintaktprozessor Eintaktprozessor in letzter Zeit vermehrt zu Programmabstürzen. Der Befehlsspeicher (Instruction Memory) sowie die Verbindungen zwischen den Blöcken konnten bereits als Fehlerquelle ausgeschlossen werden. Um das Problem zu finden, wurde der Prozessor angehalten. Folgende Signale wurden gemessen:

- **Instr[31-26] = 0x08**
- **Instr[25-21] = 0**
- **Instr[20-16] = 16**
- **Instr[15-0] = 8**
- **Registers.Read data 1 = 0**
- **Registers.Read data 2 = 24**
- **Registers.Write data = 24**
- **ALU.zero = 0**
- **ALUSrc = 0**
- **MemWrite = 0**
- **RegWrite = 1**
- **Branch = 0**

Welchen Befehl führt der Prozessor aus, als er angehalten wurde? ____ $s0, ____, ____

Betrachten Sie die gemessenen Signale. Bei welchem Signal können Sie ein Fehlverhalten beobachten? _____

Welcher Teil des abgebildeten Prozessors ist für dieses Fehlverhalten verantwortlich?

_____

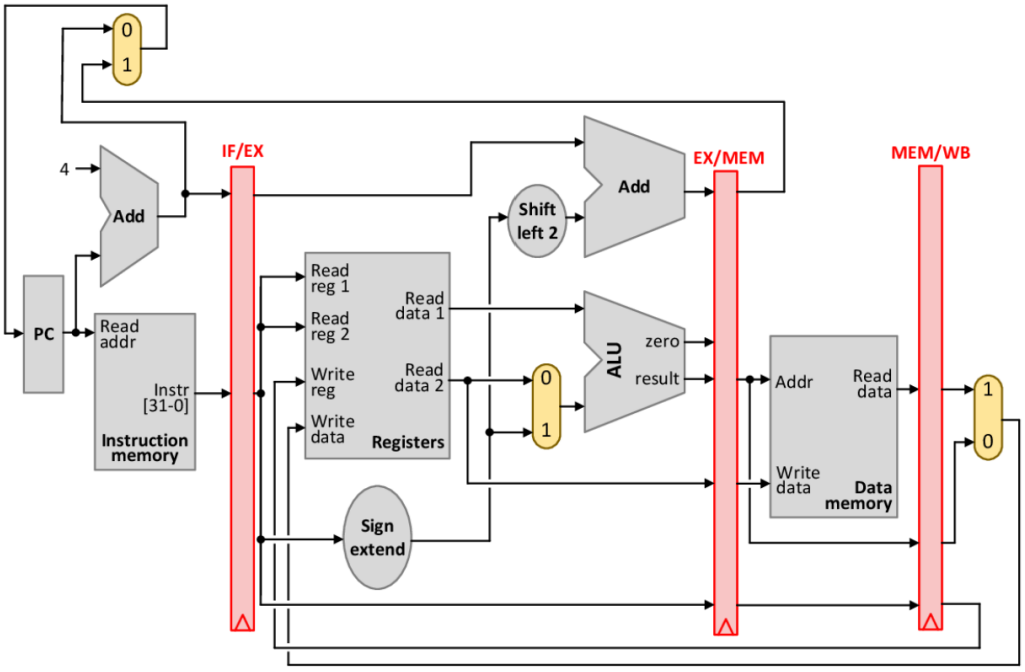
5. **Welche Aussage ist wahr?**
   - Die CPU-Leistungsgleichung $T = N_{Instr} * CPI * t_{cycle}$ lässt sich nur auf Eintaktprozessoren anwenden.
   - Der CPI-Wert gibt den Stromverbrauch einer Rechnerarchitektur an
   - Der Durchsatz (troughput) gibt an, wie viel Zeit zur Ausführung eines Programms benötigt wird.
   - Die Taktzykluszeit ist proportional zur Anzahl der ausgeführten Maschinenbefehle.
   - Die Ausführungszeit eines Programms hängt vom verwendeten Compiler ab.

6. **Die Programme P_1 und P_2 wurden in MIPS-Maschinencode übersetzt. Im Anschluss wurde ihre Ausführungszeit auf den MIPS-Prozessor M_1, M_2, M_3 und M_4 gemessen. Vervollständigen Sie die Tabelle um die gewichteten arithmetischen Mittelwerte der Ausführungszeiten. Auf Basis der Ausführungshäufigkeit soll P_1 mit 80% gewichtet werden, P_2 mit 20%. Gehen Sie davon aus, dass bei der Ausführung des gleichen Programms auf unterschiedliche Prozessoren die gleiche Befehlsfolge ausgeführt wird.**

	Prozessor M_1	**Prozessor M_2**	**Prozessor M_3**	**Prozessor M_4**
Taktfrequenz	1,0 GHz	1,0 GHz	1,2 GHz	1,5 GHz
Ausführungszeit P_1	40 s	40 s	25 s	20 s
Ausführungszeit P_2	35 s	10 s	15 s	20 s
Gewichtetes arithmetisches Mittel der Ausführungszeit	___ s	___ s	___ s	___ s

Welche Aussage lässt sich über die CPI-Werte von M_2 und M_3 bei der Ausführung von P_2 treffen? _____

7. **Die Stufen ID (instruction decode) und EX (execute) des bekannten MIPS-Pipelined-Prozessors wurden hier zusammengefasst, sodass der Prozessor nur noch vier Pipelinestufen aufweist. Eine Forwarding-Einheit oder Hazard Detection Unit sind nicht vorhanden! (Wie im bekannten fünfstufigen MIPS-Prozessor gilt: Wenn ein Register im selben Taktzyklus geschrieben und gelesen wird, gibt der Register-Block den neu geschrieben Wert aus.)**



Folgender Assemblercode soll ausgeführt werden:

```
1  addi $s5, $s5, 1
2  add $s5, $s5, $s2
3  sub $s0, $sl, $s2
4  andi $sl, $s0, 42
5  sub $s5, $s5, $s0
```

Geben Sie an, wie viele nop-Befehle jeweils zwischen den Befehlen eingefügt werden müssen, um Datenkonflikte zu vermeiden und die korrekte Ausführung sicherzustellen. Verwenden Sie nicht mehr nop-Befehle als minimal benötigt!

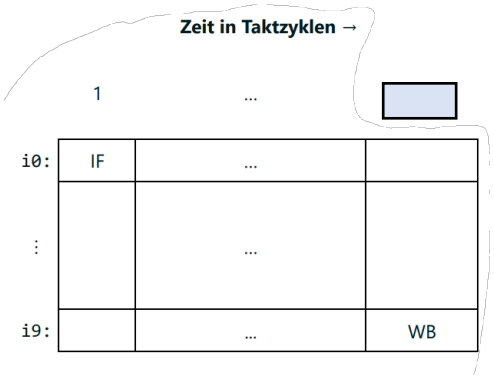
Zeilennr.	Befehl
1	addi $s5, $s5, 1
2	add $s5, $s5, $s2
3	sub $s0, $s1, $s2
4	andi $s1, $s0, 42
5	sub $s5, $s5, $s0

Durch nops lassen sich die Datenkonflikte zwar vermeiden, allerdings kann das Programm durch die zusätzlichen Befehle langsamer werden. Ist es in diesem Fall möglich, durch den geschickten Tausch zweier Instruktionen die Funktionalität des Assembler-Codes beizubehalten und auf nops gänzlich zu verzichten? Falls ja, welche Befehle müssen getauscht werden? _____

8. **Folgender Assemblercode wird auf dem aus der Übung bekannten MIPS-Prozessor mit fünf Pipeline-Stufen, Forwarding Unit und Hazard Detection Unit ausgeführt. In diesem bekannten Prozessor werden Branches immer als nicht genommen (not taken) vorhergesagt; BranchEntscheidungen werden in der ID-Stufe getroffen. Vor Ausführung enthält Register $a0 den Wert 7, $a1 den Wert 3.**

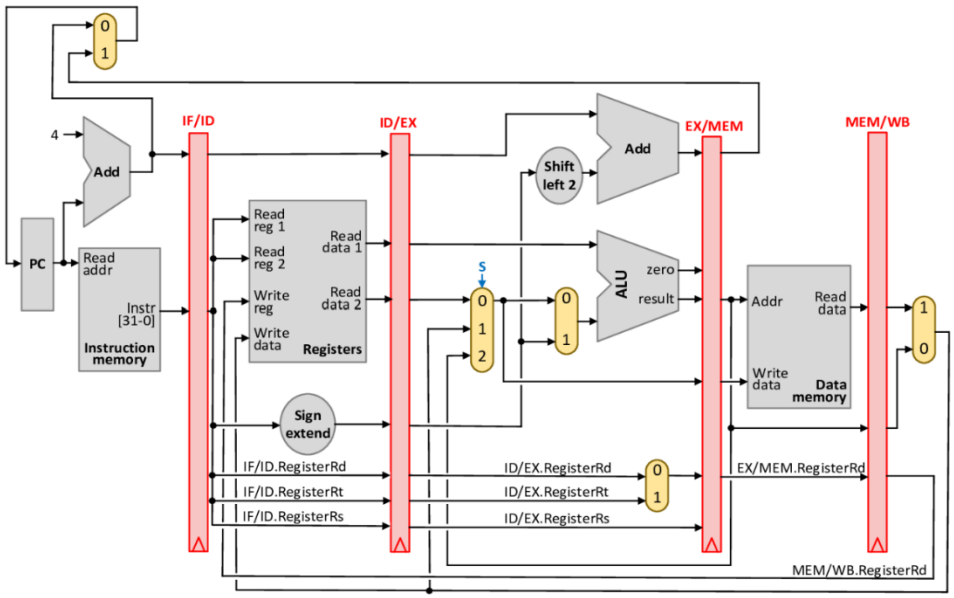
```
i0: addi $t2, $zero, 7
i1: addi $a1, $a1, 1
i2: beq $a0, $t2, i6
i3: sll $t0, $a1, 1
i4: ori $t0, $t0, 16
i5: andi $t0, $t0, 31
i6: bne $a0, $t2, i9
i7: andi $t0, $a1, 1
i8: add $t0, $t0, $a0
i9: srl $v0, $t0, 1
```

In Taktzyklus 1 durchläuft Befehl i0 die Instruction-Fetch-Stufe (IF). In welchem Taktzyklus durchläuft Befehl i9 die Write-Back-Stufe (WB)? Tragen Sie die Lösung in das dafür vorgesehene Feld in der Zeitachse ein.



9. **Welche Aussage ist wahr?**
   - Pipelining ist erforderlich, um Unterfunktionsaufrufe (function calls) zu ermöglichen.
   - Bevor ein Befehl vollständig abgearbeitet ist, kann der Pipelined-Prozessor bereits mit der Ausführung des darauf folgenden Befehls beginnen
   - Durch Pipelining können die im Eintaktprozessor auftretenden Daten- und Steuerkonflikte aufgelöst werden.
   - Der fünfstufige Pipelined-Prozessor weist einen CPI-Wert von 5 auf
   - Ein Eintaktprozessor kann typischerweise bei einer höheren Taktfrequenz betrieben werden als ein vergleichbarer Pipelined-Prozessor.

10. In dem abgebildeten MIPS-Pipelined-Prozessor wurde ein Multiplexer in die EX-Stufe eingefügt, um bestimmte Forwards zu ermöglichen.



Das Multiplexer-Steuersignal S kann die Werte 0, 1 oder 2 annehmen. Es soll durch eine Forwarding-Einheit generiert werden. Welche Bedingung ist notwendig für S=1?

- MEM/WB.RegisterRd = ID/EX.RegisterRt
- ID/EX.RegisterRt = EX/MEM.RegisterRd
- ID/EX.RegisterRs = ID/EX.RegisterRt
- MEM/WB.RegisterRd = ID/EX.RegisterRs
- MEM/WB.RegisterRd = EX/MEM.RegisterRd
- IF/ID.RegisterRd = ID/EX.RegisterRs

11. Ein 4-fach satzassoziativer Cache mit 32 Bit breiten Byte-Adressen wird genutzt, um die DRAM-Hauptspeicherzugriffe eines MIPS-Prozessors zu beschleunigen. Die Adressaufteilung ist:

- **Bits 31 – 5: Tag**
- **Bit 4: Index**
- **Bits 3 – 0: Block-Offset**

Eine Fehlzugriffsrate (miss rate) von 5% wurde gemessen. Die Trefferzeit (hit time) beträgt 15ns, die Fehlzugriffszeit (miss penalty) 100ns.

Es können maximal ___ Blöcke im Cache gespeichert werden. Die Blockgröße beträgt ___ Bits. Neben Daten- und Valid-Bits muss auch ein Tag-Speicher implementiert werden. Es sind insgesamt ___ Bits Tag-Speicher erforderlich. Die durchschnittliche Speicherzugriffszeit beträgt ___ns. Die Fehlzugriffszeit lässt sich durch _____ reduzieren.

12. Gegeben ist ein direkt abgebildeter Cache mit 4-Bit Index und einer Blockgröße von 16 Bytes unter Verwendung von Byte-Adressierung. Ausgehend von einem leeren Cache soll eine Reihe von Lesezugriffen durchgeführt werden. Vervollständigen Sie in der Tabelle die Hit/Miss-Spalte:

Zeitpunkt	Speicheradresse	Blockadresse	Index	Block-offste	Hit / Miss
1	288	18	2	0	Miss
2	112	7	7	0	_____
3	292	18	2	4	_____
4	120	7	7	8	_____
5	552	34	2	8	_____

Lösungen

Frage 1:

Assemblercode	Adresse	Maschinencode			
L1:					
andi $t0, $a0, 15	50	opcode = 12	rs = 4	rt = 8	imm = 15
addi $t1, $t0, -12	54	opcode = 8	rs = 8	rt = 9	imm = -12
beg $t1, $zero, L2	58	opcode = 4	rs = 0	rt = 9	imm = 2
addi $t1, $t0, -14	62	opcode = 8	rs = 8	rt = 9	imm = -14
addi $v0, $v0, 1	66	opcode = 8	rs = 2	rt = 2	imm = 1
L2:					
addi $v0, $v0, -2	70	opcode = 8	rs = 2	rt = 2	imm = -2
L3:					
addi $v0, $v0, 1	74	opcode = 8	rs = 2	rt = 2	imm = 1
srl $a0, $a0, 1	78	opcode = 0	rs = 0	rt = 4	rd = 4|shamt = 1|funct = 2
bne $a0, $zero, L1	82	opcode = 5	rs = 0	rt = 4	imm = -9

Frage 2:

Assemblercode	Adresse	Maschinencode					
andi $t0, $t0, 15	1000	opcode = 12	rs = 8	rt = 8	imm = 15		
srl $t2, $t0, 3	1004	opcode = 0	rs = 0	rt = 8	rd =10	shamt = 3	funct = 2
ori $t0, $a2, 1	1008	opcode = 13	rs = 6	rt = 8	imm = 1		
or $t2, $t0, $t2	1012	opcode = 0	rs = 8	rt = 10	rd = 10	shamt = 0	funct = 37
sll $t1, $t2, 10	1016	opcode = 0	rs = 0	rt = 10	rd = 9	shamt = 10	funct = 0

Frage 3:

Baustein „4“ soll an die Stelle „P“ eingefügt werden.

Das Fragezeichen im eingefügten Baustein wird mit „valRtZero“ verbunden.

newSig muss von dem Block „Control“ erzeugt werden.

Die Unterscheidung zum addi-Befehl erfolgt mittels „Opcode“.

Falls ein aiuz-Befehl erkannt wird, muss newSig den Wert „1“ annehmen.

Frage 4:

addi $s0, $zer0, 8

ALUSrc

Control

Frage 5:

Die Ausführungszeit eines Programms hängt vom verwendeten Compiler ab.

Frage 6:

M_1 = 39s M_2 = 34s M_3 = 23s M_4 = 20s

CPI(M2) < CPI(M3)

Frage 7:

Zeilennr.	Befehl
1	addi $s5, $s5, 1
	1x nop
2	add $s5, $s5, $s2
	kein nop
3	sub $s0, $s1, $s2
	kein nop
4	andi $s1, $s0, 42
	1x nop
5	sub $s5, $s5, $s0

Frage 8:

12

Frage 9:

Bevor ein Befehl vollständig abgearbeitet ist, kann der Pipelined-Prozessor bereits mit der Ausführung des darauf folgenden Befehls beginnen.

Frage 10:

MEM/WB.RegisterRd = ID/EX.RegisterRt

Frage 11:

maximal „8“ Blöcke

Blockgröße „128“

„216“ Bits Tag-Speicher

durchschnittliche Speicherzugriffszeit „20ns“

Fehlzugriffszeit reduzieren durch „Reduktion der DRAM-Latenz“

Frage 12:

Zeitpunkt 2 = Miss

Zeitpunkt 3 = Hit

Zeitpunkt 4 = Hit

Zeitpunkt 5 = Miss