

HTBLA Hollabrunn - Höhere Abteilung für Elektronik - Technische Informatik

Datum: 20.01.2016	2. Test	HWE	Aufgabe:	Wertung:	Bewertung:
Klasse: 5BHEL	Lehrer:	DUM Grp B	1	25	25
Note:	Wertungsskala:		2	15	5+5+2
	0 - 50 Nicht genügend		3	28	22
	51 - 62 Genügend		4	32	30
	63 - 75 Befriedigend				
	76 - 89 Gut				
	90 - 100 Sehr gut				
Zeit: max. 15" + Vorber.	Schüler: Nötz Florian		Summe:	100	89

1) Ein RISC-Prozessor soll eine 5-stufige Phasenpipeline erhalten.

a) (5P) Welche Stufen wird diese sinnvollerweise haben? (Bezeichnung entsprechend dem Hauptgeschehen in der jeweiligen Stufe bitte ausschreiben)

b) (5P) Angenommen es gäbe keine Pipeline-Hazards. Wieviele Taktzyklen werden im Idealfall benötigt, um ein lineares Programm bestehend aus 10 Befehlen in der 5-stufigen Pipeline fertigzustellen? (nehmen Sie an, dass die Pipeline beim Programmstart leer war).

c) (15P) Benennen Sie die drei wichtigsten Pipeline-Hazards und erläutern Sie, wie jeweils mit einer bestimmten Maßnahme Abhilfe geschaffen werden kann.

2) a) (5P) Was versteht man unter einer „superskalaren Prozessorarchitektur“

b) (5P) Vergleichen Sie „in order execution“ mit einer „out of order execution engine“ und argumentieren Sie, warum nahezu alle moderne Superskalarp Prozessoren „out of order“ arbeiten.

c) (5P) Welches besonderes Problem ergibt sich bei „out of order execution engines“ und wie wird dieses gelöst?

1a) Instruction Decode, Instruction Fetch, Instruction Decode, Execute, Memory, Write back

1b) 5 Takte (5 + 9 = 14 Takte)

5
6
7
8
9
10
11
12
13
14

1c) Control Hazard: Branch Prediction (Sprünge vorhersagen)

Data Hazard: Forwarding (Daten von z.B. Execute in ID-Phase

Structural Hazard: ~~Struktur~~ Mehr Hardware (z.B. mehrere ALUs, FloatingPoint Unit, ...) "voranschicken"

2a) Prozessor mit mehreren parallelen Phasenpipelines

2b) in order: alle Befehle werden nacheinander in die Pipeline geschoben
→ kann zu kritischen Hazards führen (z.B. Structural, nicht alle Pipelines besitzen FloatingPoint Unit, ...)

out of order: Befehle werden dynamisch einer bestimmten Pipeline zugeführt
→ Reihenfolge wird verändert (deswegen am Ende der Pipeline wieder „zurücksortieren“), Ressourcen werden best- und schnellstmöglich genutzt

Rey. Nennung
Complete

Bei out of order: stark anfällig für Data/Control Hazards, ~~Stütz~~ Hilfe bringt Dynamic Branch Prediction, Branch Target & ALU

3) **Zeigen Sie**, wie der angegebene Befehl in der CPU entsprechend der dargestellten Blockdarstellung in einem Taktzyklus ausgeführt werden kann. **Gr A:** LW \$09, 0x12 (\$01) **GrB:** SW \$07, 0x01 (\$03)
 Der Befehl steht auf der Programmspeicherstelle 0x A0.

a) **Beschriften sie alle relevanten Leitungsbündel** mit den in ihnen transportierten Werten (in Hex), alle nichtbeschrifteten Leitungen seien don't care.

c) **Geben Sie die Werte (0, 1) aller relevanten Steuersignale** an (alle nichtbeschrifteten Leitungen seien don't care).

Achtung: alle Zahlen bei dieser Aufgabe sind positive Hexadezimalwerte; führende Nullen sollen beim Anschreiben weggelassen werden.

Angaben:

Die Register sollen folgende Inhalte haben:

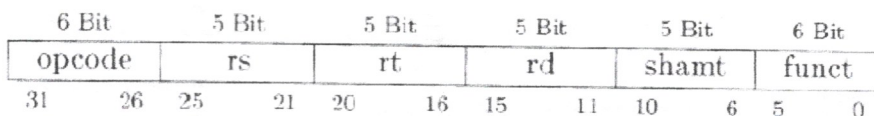
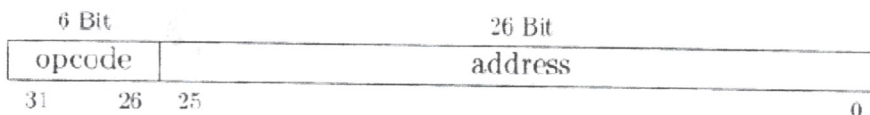
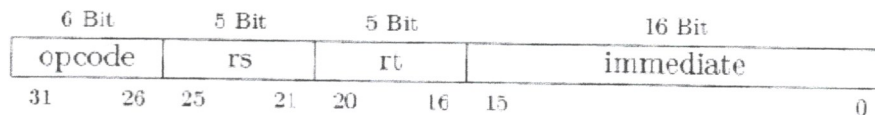
Reg	Wert	Reg	Wert	Reg	Wert	Reg	Wert
\$00	0x0	\$02	0x24	\$07	0x88	\$0c	0xc0
\$01	0x10	\$03	0x24	\$09	0x999	\$1f	0xd2

Im Datenspeicher stehen folgende Werte:

Adr	Inh	Adr	Inh	Adr	Inh	Adr	Inh
0x00	0x01	0x22	0x89	0x6a		0x74	
0x03	0x33	0x25	0x02	0x6d		0x77	

LW rt, Im16 (rs) | Load Word $R[rt] := \text{Mem} [R[rs] + \text{Im16}]$
 SW rt, Im16 (rs) | Store Word $\text{Mem} [R[rs] + \text{Im16}] := R[rt]$

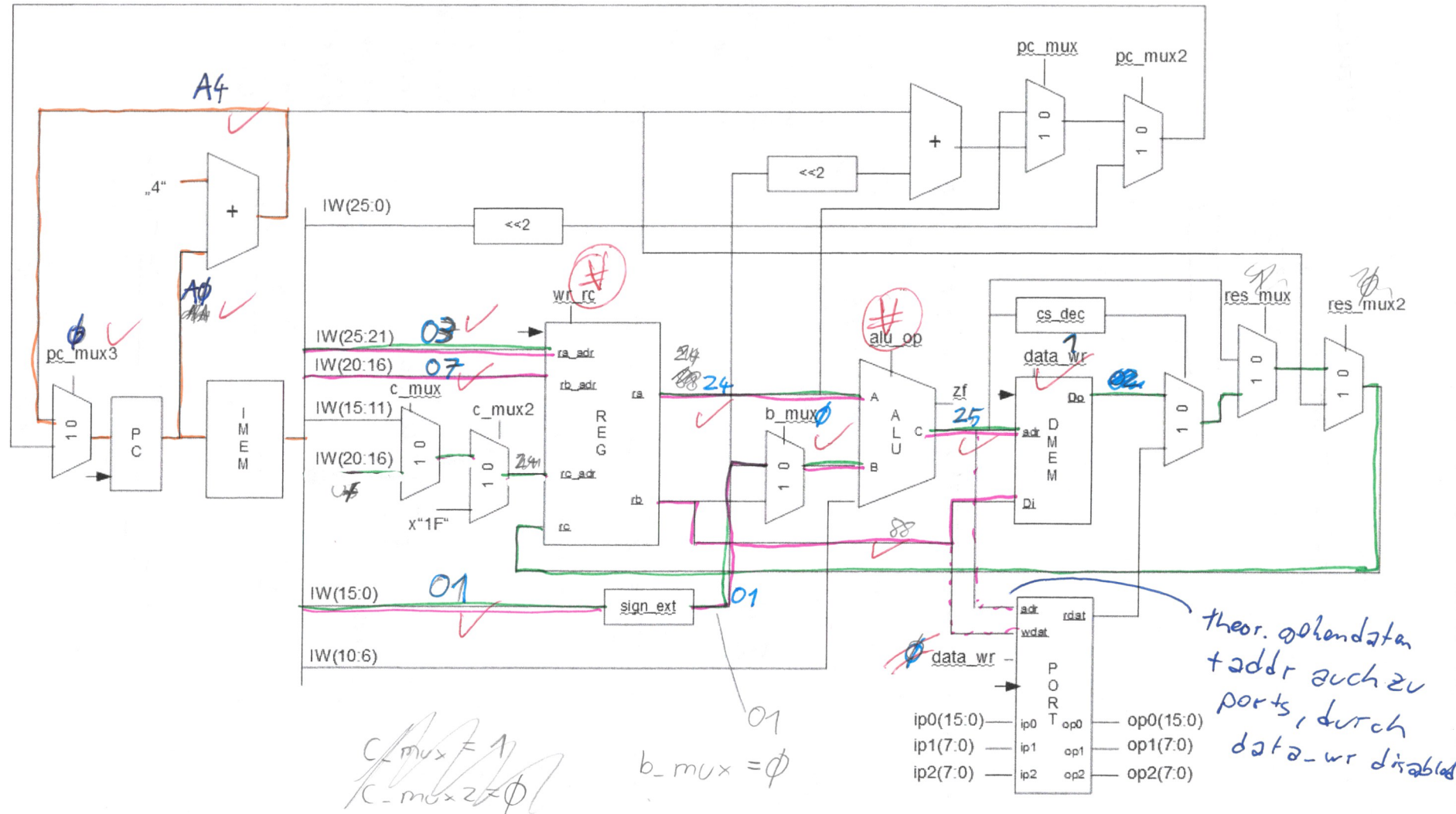
Befehlsformate der MIPS-Architektur:



0x A0
 AB
 AC
 AD
 AE

0x A0
 A1
 A2
 A3
 A4

R

 $3F$ 

4) (je 4 P) Ein Rechner soll einen direktabbildenden Cache besitzen. Bearbeiten Sie die folgenden Aufgaben bzw. beantworten Sie die gestellten Fragen (je 4 P).

Angaben:

- Hauptspeichergröße: 2 MiB = $2^{20} \cdot 2 = 2^{21}$ Byte
- Zugriffsbreite zu Speicher = Breite einer Cacheline: 16 Byte = 2^4 Byte
- der Datenbereich des Cache soll 4 kiB groß sein = $2^2 \cdot 2^{10} = 2^{12}$ Byte

$$\# \text{Cachelines} = \frac{2^{12}}{2^4} = 2^8$$

- ✓ a) geben Sie die niedrigste und die höchste Hauptspeicheradresse an (hex).
 ✓ b) Zerteilen Sie die Adresse in die für der Cachecontroller wichtigen Teile und benennen Sie diese; verwenden Sie als Beispiel die Hauptspeicheradresse: GRA: 0x 2 4 6 8 1 Gr B: 0x 5 0 3 4 F
 ✓ c) Berechnen Sie, wieviele Cachelines der Cache umfasst (Dezimal)
 ✓ d) Welche Größe hat der Tagbereich des Caches in BIT (Dezimal) (+Skizze)
 ✓ e) nehmen Sie an, dass auf der angegebenen Hauptspeicheradresse der Wert: 0x 44 steht; in welcher Cacheline (Nummer in Hex) steht dieses Byte, falls der Zugriff einen Cache Hit ergibt? (+ Skizze Datenbereich)
 1/2 f) Was steht dabei im Tag-Bereich im Eintrag mit dieser Nummer? (+ Skizze Tagbereich)
 ✓ g) Was muss der Cache-controller bei einem Cache-Miss durchführen.
 ✓ h) Beschreiben Sie den Unterschied zwischen write-back und write through Cache-Strategie?

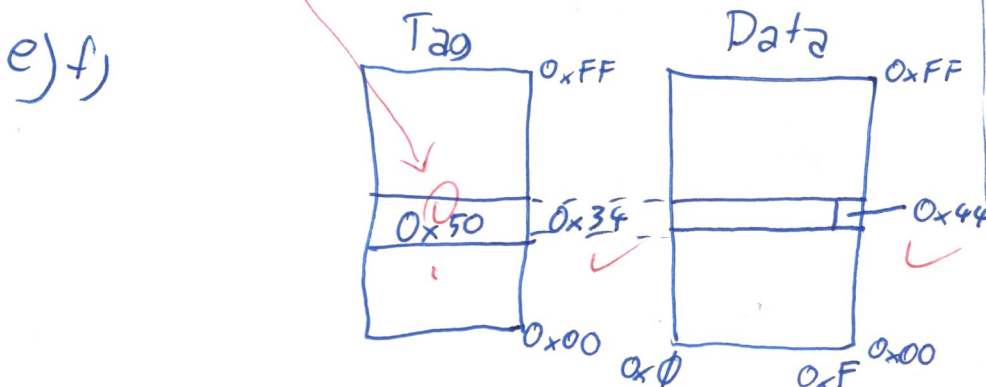
a) 21 Bit Adresse = 0x 00000000 ... 0x 1 F F F F F

b) Byte Offset = 0xF
 Index = 0x34
 Tag = 0x050

0x 50:34:F
 Tag Ind Off

h) write-back: Wenn Daten im Cache manipuliert werden, werden Sie mit dem Dirty Bit versehen, bleiben aber so lange im Cache, bis die Line für andere Daten benötigt wird (oder Cache-Flush). In diesem Fall müssen die Daten dann zurückgeschrieben werden, bevor die Line neu beschrieben wird.
 write-through: die Daten werden bei manipulation direkt weiter in den Hauptspeicher (oder überliegende caches) geschrieben.

c) # Cache Lines = $\frac{2^{12}}{2^4} = 2^8 = 256$ Lines



g) Er muss die Daten von dem nächsten (=darüberliegenden) Cache oder Hauptspeicher holen (dauert länger)