Klausur Mikroprozessortechnik
03. April 2012

Name: ……. Vorname ……

Matr.-Nr: ……. Studiengang ……

Hinweise:

- Bitte füllen Sie vor dem Bearbeiten der Aufgaben das Deckblatt sorgfältig aus.
- Die Klausur besteht aus 6 doppelseitig bedruckten Blättern. Bitte Überprüfen Sie ihr Exemplar auf Vollständigkeit.
- Zur Klausur zugelassen sind ausschließlich Schreibutensilien, aber **kein Taschenrechner und kein eigenes Papier**!
- Schreiben Sie bitte auf alle Zusatzblätter Ihren Namen und Ihre Matrikelnummer.
- Betrugsversuche führen zum **sofortigen** Ausschluss der Klausur.
- Lösungen in Bleistift können nicht gewertet werden.
- Voraussetzung für die volle Punktzahl ist immer, dass der Lösungsweg vollständig erkennbar ist.
- Die Bearbeitungszeit beträgt **90** Minuten.

Viel Erfolg!

AUFGABE	PUNKTE
1	
2	
3	
4	
Gesamtpunkte	
Note	

Aufgabe 1) Allgemeines (6 Punkte)

1.1) Welche IEEE-754-Gleitkommazahl wird durch folgendes 32-Bit Wort dargestellt? **(1 Punkt)**

11000010101000000000000000000000

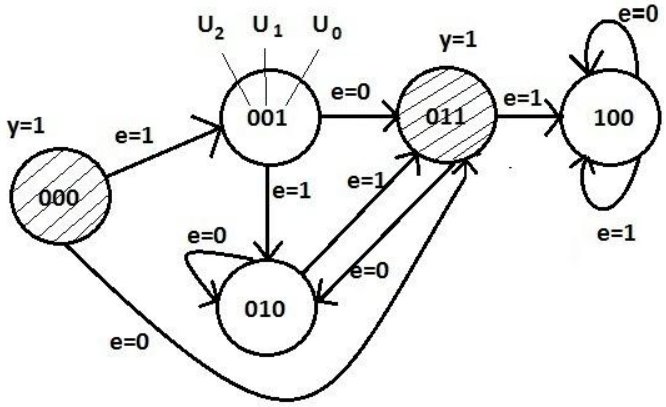
1.2) Entwerfen Sie ein minimales Schaltnetz, das erkennt, ob eine 3-Bit Zahl ($x_2 x_1 x_0$) im Zweierkomplement größer als -3 ist. **(2 Punkte)**

1.3) Nennen Sie die 3 Cacheorganisationsformen aus der Vorlesung. Beschreiben Sie kurz, wofür Ersetzungsstrategien gebraucht werden **(2 Punkte)**

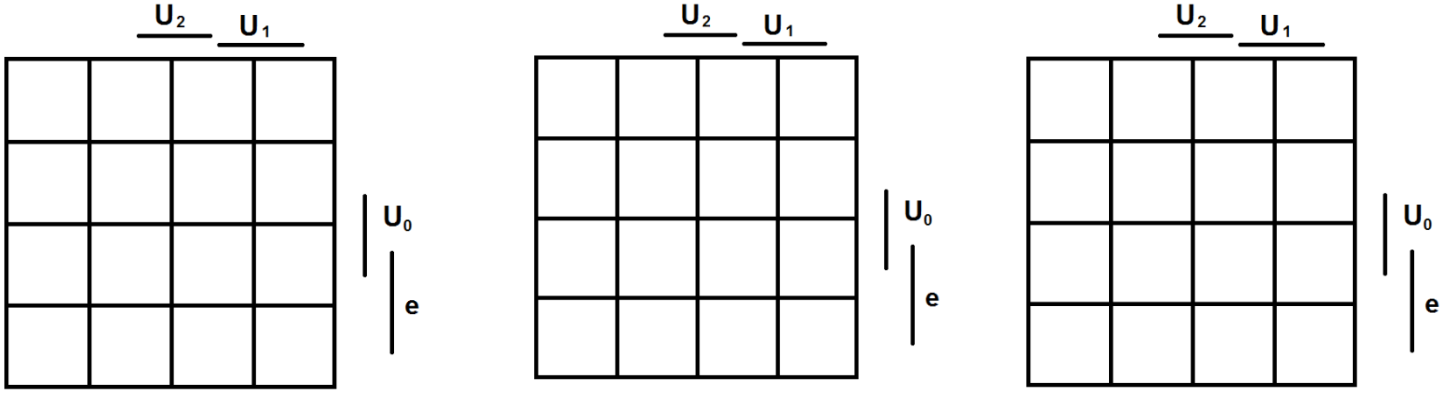
1.4) Was versteht man unter einem 3-Adressrechner? **(1 Punkt)**

Aufgabe 2) Schaltwerksentwurf (6 Punkte)

Gegeben sei das folgende Zustandsdiagramm, mit dem Eingang *e* und dem Ausgang *y*.



2.1) Stellen Sie die minimalen Übergangsfunktionen auf! Nutzen Sie dazu die vorgegebenen KV Tafeln! **(1.5 Punkte)**



2.2) Bestimmen Sie die entsprechenden Beschaltungsfunktionen. Nutzen Sie für das Zustandsbit U_2 ein JK-FlipFlop, für das Zustandsbit U_1ein SR-FlipFlop und für das Zustandsbit U_0 ein D-FlipFlop **(2.5 Punkte)**

2.3) Bestimmen Sie die minimale Ausgansfunktion für y, die nur in den schraffierten Zuständen den Wert 1 annimmt. **(1 Punkt)**

2.4) Zeichnen Sie ein taktzustandsgesteuertes D-FlipFlop auf Gatterebene. **(1 Punkt)**

Aufgabe 3) Datenpfadentwurf (10 Punkte)

Beziehen Sie sich in dieser Aufgabe auf den MIPS Mehrzyklendatenpfad wie in Abb. 2.1 dargestellt.

3.1) Beschreiben Sie kurz die Funktion des Steuersignals IorD **(1 Punkt).**

3.2) Wie viele verschiedene Operationen könnte die ALU aus dem MIPS Mehrzyklendatenpfad umsetzen? **(0.5 Punkte)**

3.3) Nennen Sie die 5 Befehlszyklen des MIPS-Mehrzyklendatenpfades **(2.5 Punkte)**

3.4) Betrachten Sie den folgenden Codeausschnitt eines MIPS-Assemblerprogrammes.

```
0x000003FC      loop:  addi   $sp,$sp,4
0x00000400             li     $s0,4
0x00000404             ori    $t3,$zero,0x5
0x00000408             beq    $s0,$t3,loop
0x0000040C             sw     $s0,0($sp)
0x00000410             lw     $t3,20($t0)
0x00000414             add    $t0,$t0,$t3
```

Übersetzen Sie den Verzweigungsbefehl an Adresse 0x408 in die MIPS-Maschinensprache. Nutzen Sie gegebenenfalls die Tabellen im Anhang zu Hilfe. **(1 Punkt)**

3.5) Für die richtige Befehlsabarbeitung im vorliegenden Datenpfad werden vom Steuerwerk die entsprechenden Steuersignale gesetzt. Handelt es sich bei dem Steuerwerk um ein Schaltnetz oder ein Schaltwerk? Begründen Sie! **(1 Punkt)**

3.6) In der folgenden Tabelle sollen verschiedene Steuersignale des MIPS Mehrzyklen-datenpfades in den verschiedenen Taktzuständen untersucht werden. Jede Zeile bezieht sich auf ein bestimmtes Steuersignal, dessen Belegungen in den verschiedenen Takten für einen bestimmten Befehl festgelegt werden sollen. Ist kein Befehl eingetragen, soll ein passender angegeben werden. Berücksichtigen Sie auch explizit Don't Care Fälle! Ist ein Befehl in weniger Takten abgearbeitet, kennzeichnen Sie das durch zwei Striche ´--´ in den entsprechenden letzten Taktspalten. **(4 Punkte)**

Steuersignal	Befehl	1. Takt	2. Takt	3. Takt	4. Takt	5. Takt
RegDst	add					
PCWrite				1		
RegWrite	lw					
MemtoReg	sw					
ALUOp	lw					
ALUSrcA	ori					
ALUSrcB	beq					
ALU Steuervektor				0110	****	

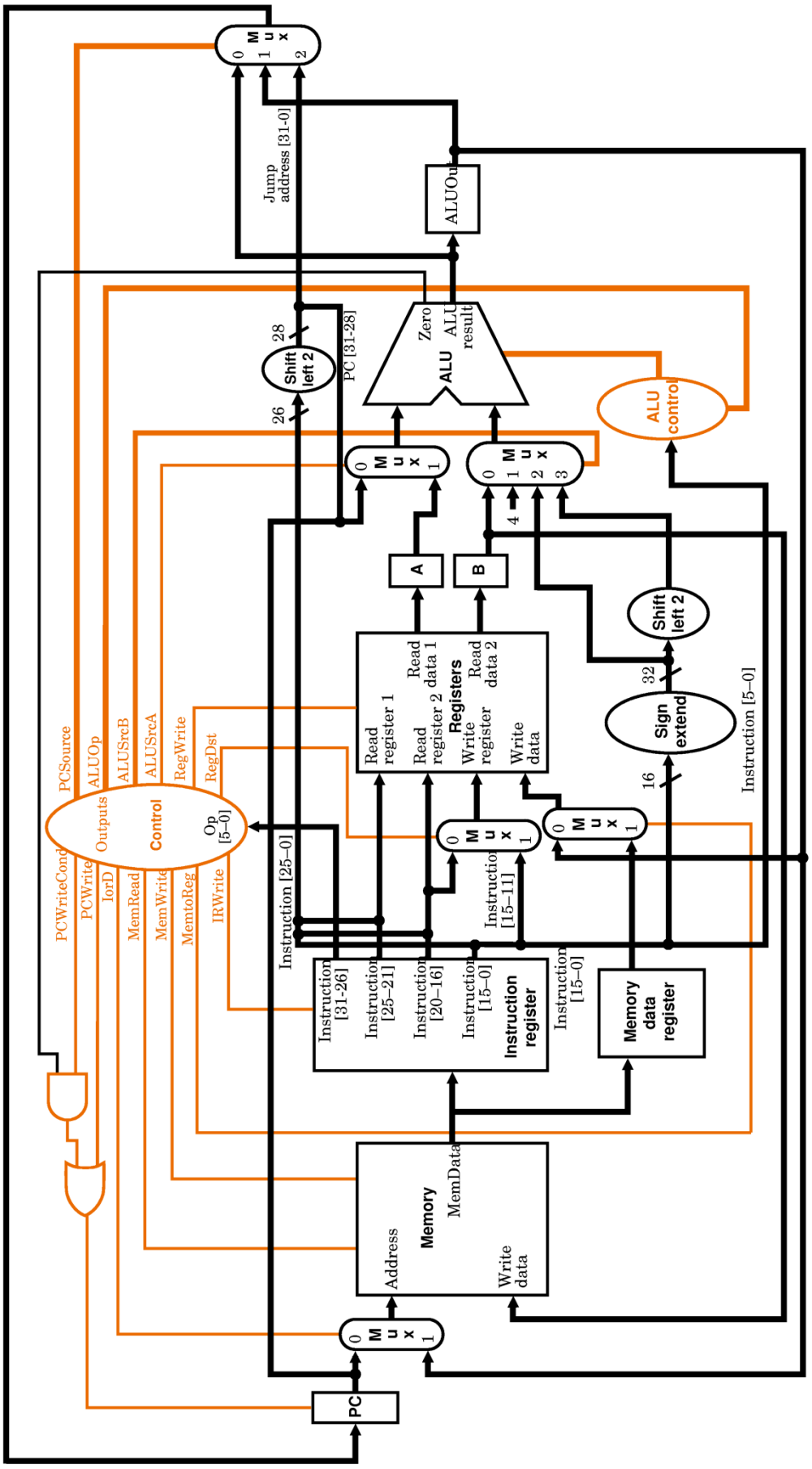


Abbildung 2.1 – MIPS Mehrzyklendatenpfad

Aufgabe 4) Assembler (8 Punkte)

Eine natürliche Zahl n wird vollkommene Zahl (auch perfekte Zahl) genannt, wenn sie die Summe aller ihrer positiven Teiler außer sich selbst ist.

Beispiele für vollkommene Zahlen:
6 = 1 + 2 + 3 28 = 1 + 2 + 4 + 7 + 14

In der folgenden Aufgabe soll nun ein Programm in MIPS-Assembler geschrieben werden, dass überprüft, ob eine Zahl eine vollkommene Zahl ist oder nicht.

In dieser Aufgabe soll davon ausgegangen werden, dass die zugrundeliegende MIPS Architektur über **keine Multiplikations- und Divisionsbefehle** verfügt!

Halten Sie alle Konventionen für MIPS Unterprogrammaufrufe ein. Kommentieren Sie jede Zeile!

4.1) Schreiben Sie zunächst eine Funktion „*_isDivisor*“. Diese Funktion bekommt **zwei positive Zahlen** übergeben, einen Dividend und einen Divisor. Es soll nun überprüft werden, ob der Divisor ein ganzzahliger Teiler des Dividenden ist. Ist das der Fall, soll der Wert des Divisors wieder zurückgegeben werden, andernfalls eine Null. **(3 Punkte)**

4.2) Schreiben Sie nun eine Funktion „*_isPerfectNumber*“. Diese Funktion bekommt eine **positive Zahl** größer als 1 übergeben und soll eine 1 zurückliefern, wenn es sich um eine „vollkommene Zahl“ handelt beziehungsweise eine Null falls nicht. Nutzen Sie die Funktion aus Aufgabenteil a)! **(4 Punkte)**

Hinweis: Bedenken Sie, dass hier eine Funktion in einer Funktion aufgerufen wird! Treffen Sie alle nötigen Vorkehrungen!

4.3) Damit das Programm auf dem MIPS Prozessor ausgeführt werden kann, muss es in die MIPS Maschinensprache übersetzt werden. Erläutern Sie kurz den Unterschied zwischen einem Compiler und einem Assembler. **(1 Punkt)**

MIPS Befehlsreferenz

add	add $s1,$s2,$s3	$s1 = $s2 + $s3
subtract	sub $s1,$s2,$s3	$s1 = $s2 - $s3
add immediate	addi $s1,$s2,100	$s1 = $s2 + 100
add unsigned	addu $s1,$s2,$s3	$s1 = $s2 + $s3
subtract unsigned	subu $s1,$s2,$s3	$s1 = $s2 - $s3
add immediate unsigned	addiu $s1,$s2,100	$s1 = $s2 + 100
move from coprocessor register	mfc0 $s1, $epc	$s1 = $epc
load word	lw $s1, 100($s2)	$s1 = Memory[$s2 + 100]
store word	sw $s1, 100($s2)	Memory[$s2 + 100] = $s1
load half unsigned	lhu $s1, 100($s2)	$s1 = Memory[$s2 + 100]
store half	sh $s1, 100($s2)	Memory[$s2 +100] = $s1
load byte unsigned	lbu $s1,100($s2)	$s1 = Memory[$s2 + 100]
store byte	sb $s1,100($s2)	Memory[$s2 + 100] = $s1
load upper immediate	lui $s1, 100	$s1 = 100 * 2^16
and	and $s1,$s2,$s3	$s1 = $s2 & $s3
or	or $s1,$s2,$s3	$s1 = $s2 | $s3
nor	nor $s1,$s2,$s3	$s1 = ~($s2 | $s3)
and immediate	andi $s1,$s2,100	$s1 = $s2 & 100
or immediate	ori $s1,$s2,100	$s1 = $s2 | 100
shift left logical	sll $s1,$s2,10	$s1 = $s2 << 10
shift right logical	srl $s1,$s2,10	$s1 = $s2 >> 10
branch on equal	beq $s1,$s2,25	if ($s1==$s2) GoTo PC+4+100
branch on not equal	bne $s1,$s2,25	if($s1!= $s2) GoTo PC+4+100
branch on greater equal	bge $s1,$s2,25	if($s1>= $s2) GoTo PC+4+100
branch on greater than	bgt $s1,$s2,25	if($s1> $s2) GoTo PC+4+100
branch on less equal	ble $s1,$s2,25	if($s1<= $s2) GoTo PC+4+100
branch on less	blt $s1,$s2,25	if($s1< $s2) GoTo PC+4+100
set on less than	slt $s1,$s2,$s3	if($s2<$s3) $s1=1 else $s1=0
set less than immediate	slti $s1,$s2,100	if($s2<100) $s1=1 else $s1=0
set less than unsigned	sltu $s1,$s2,$s3	if($s2<$s3) $s1=1 else $s1=0
jump	j 2500	GoTo 10000
jump register	jr $ra	GoTo $ra
jump and link	jal 2500	$ra = PC + 4, GoTo 10000

Tabelle 1 MIPS-Befehlsreferenz

Flip Flop Übergangs- und Beschaltungsfunktionen

	D – FF	SR – FF	JK – FF	T - FF
Übergangsfunktion	$u^+ = d$	$u^+ = \bar{u}s + u\bar{r}$	$u^+ = j\bar{u} + \bar{k}u$	$u^+ = \bar{t}u + t\bar{u}$
Beschaltungsfunktion	$d = u^+$	$s = u^+\|_{u=0}$ $r = \overline{u^+}\|_{u=1}$	$j = u^+\|_{u=0}$ $k = \overline{u^+}\|_{u=1}$	$t = u^+ \oplus u$

Tabelle 2 Beschaltungsfunktionen

Tabelle 3 MIPS-Maschinensprache

Mnemonik	Format							Anmerkung
add	R	0	18	19	17	0	32	add $s1, $s2, $s3
sub	R	0	18	19	17	0	34	sub $s1, $s2, $s3
lw	I	35	18	17	100			lw $s1, 100($s2)
sw	I	43	18	17	100			sw $s1, 100($s2)
and	R	0	18	19	17	0	36	and $s1, $s2, $s3
or	R	0	18	19	17	0	37	or $s1, $s2, $s3
nor	R	0	18	19	17	0	39	nor $s1, $s2, $s3
andi	I	12	18	17	100			andi $s1, $s2, $s3
ori	I	13	18	17	100			ori $s1, $s2, $s3
sll	R	0	0	18	17	10	0	sll $s1,$s2,10
srl	R	0	0	18	17	10	2	srl $s1,$s2,10
beq	I	4	17	18	25			beq $s1,$s2,100
bne	I	5	17	18	25			bne $s1,$s2,100
slt	R	0	18	19	17	0	42	slt $s1, $s2, $s3
j	J	2			2500			j 10000
jr	R	0	31	0	0	0	8	jr $ra
jal	J	3			2500			jal 10000
Bitbreite		6 Bit	5 Bit	5 Bit	5 Bit	5 Bit	6 Bit	
R-Format	R	op	rs	rt	rd	shamt	funct	
I-Format	I	op	rs	rt	address			
J-Format	J	op	address					

Tabelle 4 MIPS-Register

Name	RegisterNr.	Nutzung
$zero	0	Der konstante Wert 0
$v0 - $v1	2-3	Werte für Ergebnisse und für die Auswertung
$a0 - $a3	4-7	Argumente
$t0 - $t7	8-15	Temporäre Variablen
$s0 - $s7	16-23	Gespeicherte Variablen
$t8 - $t9	24-25	Globaler Zeiger
$gp	28	Kellerzeiger
$sp	29	Rahmenzeiger
$fp	30	Rahmenzeiger
$ra	31	Rücksprungadresse

Opcode	ALUOp	Operation	Funct-field	ALU Operation	ALU Controlinput
lw	00	load word	******	Addition	0010
sw	00	store word	******	Addition	0010
beq	01	branch on equal	******	Subtraction	0110
bne	01	branch not equal	******	Subtraction	0110
R-Command	10	add	100000	Addition	0010
R-Command	10	subtract	100010	Subtraction	0110
R-Command	10	and	100100	And	0000
R-Command	10	or	100101	Or	0001
R-Command	10	set on less than	101010	Less Than	0111

Tabelle 5 MIPS – ALU Steuervektoren