Klausur Mikroprozessortechnik
28. März 2011

Name: ……. Vorname ……

Matr.-Nr: ……. Studiengang ……

Hinweise:

- Bitte füllen Sie vor dem Bearbeiten der Aufgaben das Deckblatt sorgfältig aus.
- 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 strukturiert und vollständig erkennbar ist.
- Überprüfen Sie zuerst, ob die Klausur vollständig ist. (6 Blätter)
- Die Bearbeitungszeit beträgt **120** Minuten.
- **Hinweis**: Im Anhang finden Sie nützliche Formeln und Tabellen, die Ihnen für die Lösung der Aufgaben hilfreich sein könnten.

Viel Erfolg!

AUFGABE	PUNKTE
1	
2	
3	
4	
Gesamtpunkte	
Note	

Aufgabe 1) Allgemeines (5 Punkte)

a) Ordnen Sie den folgenden Elementen die richtige Logikschaltung zu.
Kennzeichnen Sie auch explizit, wenn ein Element nicht in den Schaltungsbildern vorhanden ist.
(3 Punkte)

a) 4:1 Multiplexer _____

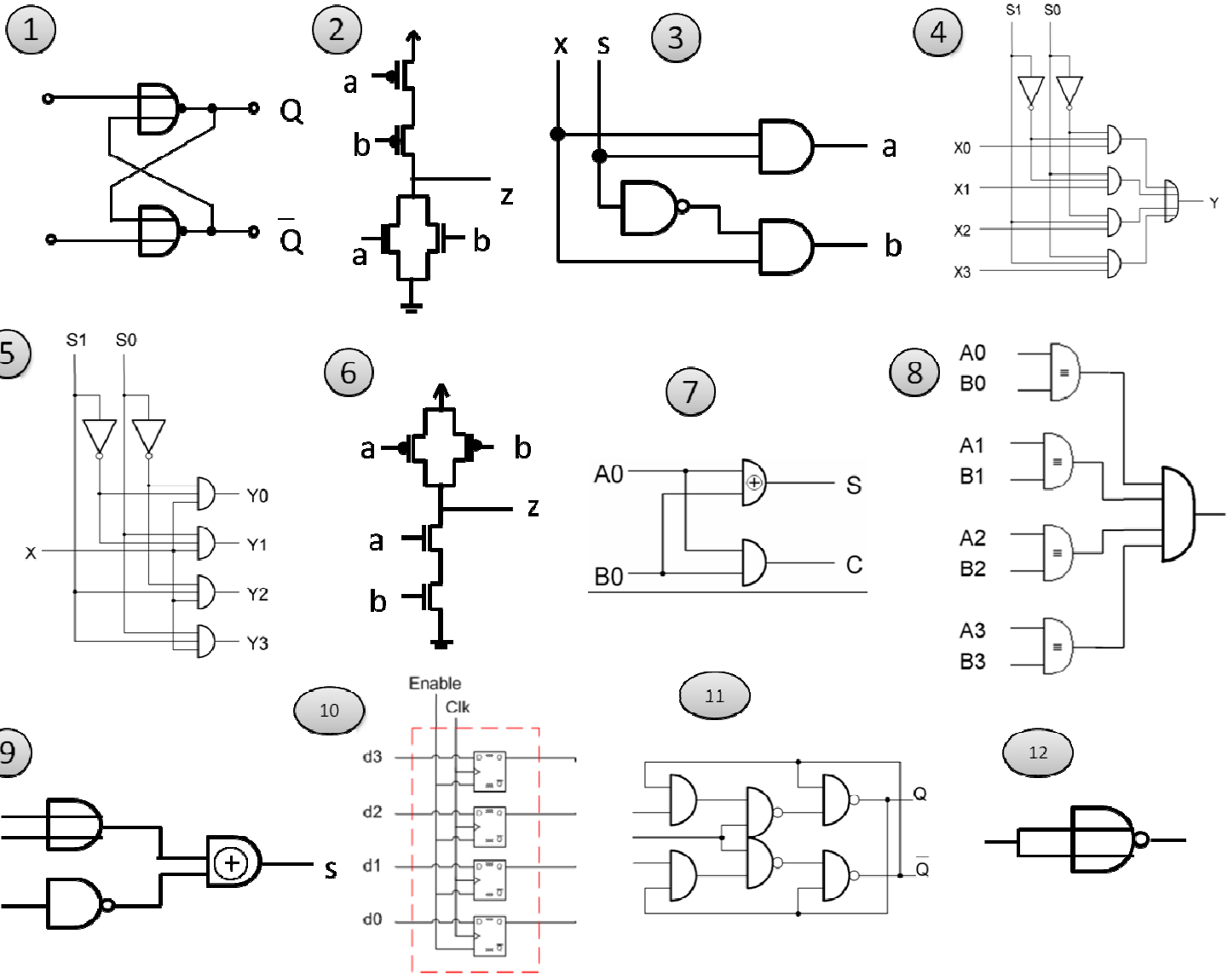
b) Halbaddierer _____

c) Volladdierer _____

d) JK-FlipFlop _____

e) NOR _____

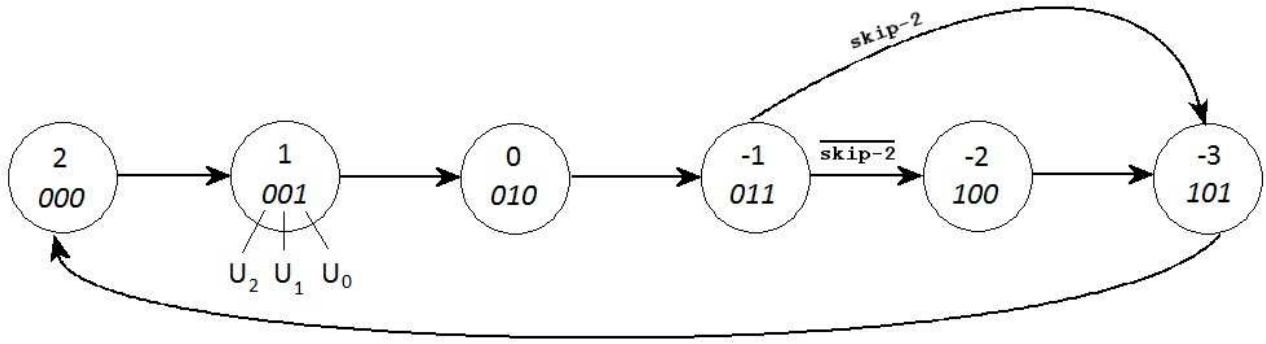
f) 4-Bit Komparator _____



b) Setzen Sie die logische Funktion $y = a \cdot \bar{b}$ mit einem CMOS-Schaltnetz um. **(2 Punkte)**

Aufgabe 2) Schaltwerksentwurf (10 Punkte)

In dieser Aufgabe soll mithilfe eines Synchronschaltwerkes ein besonderer binärer Rückwärtszähler entworfen werden. Dieser Zähler soll synchron zum Takt von 2 beginnend bis -3 herunter zählen und das Ergebnis im 3-Bit Zweierkomplement ausgeben. Bei der -3 angekommen, soll wieder von 2 gestartet werden. Zusätzlich soll es die Möglichkeit geben, die „-2“ zu überspringen, wenn das Signal „skip-2“ aktiv ist. Für die Lösung dieser Aufgabe wird Ihnen folgendes Zustandsdiagramm vorgeschlagen:



a) Wurde die Zustandscodierung in Bezug auf die Generierung der Ausgangssignale vorteilhaft gewählt? Begründen Sie. **(1 Punkt)**

b) Bestimmen Sie für das **gegebene** Zustandsdiagramm die minimalen Übergangsfunktionen. **(3 Punkte)**

c) Bestimmen Sie die Beschaltungsfunktionen für die Flipflops, mit denen das System umgesetzt wird. Verwenden Sie für das Most-Significant-Bit der Zustandscodierung ein JK-Flipflop, für das mittlere Bit ein D-Flipflop und für Least-Significant-Bit ein SR-Flipflop. **(3 Punkte)**

d) Bestimmen Sie die Werte für die minimalen Ausgangsfunktionen, um den aktuellen Zählwert korrekt im Zweierkomplement darstellen zu können. **(3 Punkte)**

Aufgabe 3) Datenpfadentwurf (8 Punkte)

Betrachten Sie für die folgenden Aufgaben den gegebenen MIPS Einzyklendatenpfad (Abbildung 1).

1.1) Beziehen Sie sich auf das Steuersignal „*RegDst*“. **(1,5 Punkte)**

Nennen Sie einen Befehl, bei dem diese Steuerleitung auf logisch „0“ gesetzt wird.

Nennen Sie einen Befehl, bei dem diese Steuerleitung auf logisch „1“ gesetzt wird.

Nennen Sie einen Befehl, bei dem es egal ist, welchen Wert diese Steuerleitung annimmt.

1.2) Der Datenpfad soll nun um den Befehl „branch register zero“ erweitert werden.
brz r1,r2

Dabei soll zu einer Adresse die im Register r2 gespeichert ist gesprungen werden, wenn der Wert von Register r1 gleich Null ist.

Erweitern Sie zunächst den Datenpfad um zusätzliche Hardwareelemente, damit dieser Befehl abgearbeitet werden kann. **(1 Punkt)**
Hinweis: Es gibt durchaus mehrere Lösungen, alles was funktioniert ist richtig!

Bestimmen Sie die optimalen Belegungen der Steuerleitungen für den Befehl brz. Nutzen Sie dafür folgende Tabelle. Falls Sie neue Steuerleitungen hinzugefügt oder bereits vorhandene Steuerleitungen modifiziert haben, passen Sie die Tabelle entsprechend an! **(3 Punkte)**

Befehl	Reg Dst	Reg Write	ALU Src	Branch	Mem Read	Mem Write	Mem ToReg	ALUOp		
brz										

In welchem Format würden Sie diesen Befehl speichern? Begründen Sie und geben Sie die einzelnen Felder an. Wählen Sie einen beliebigen Opcode. **(1 Punkt)**

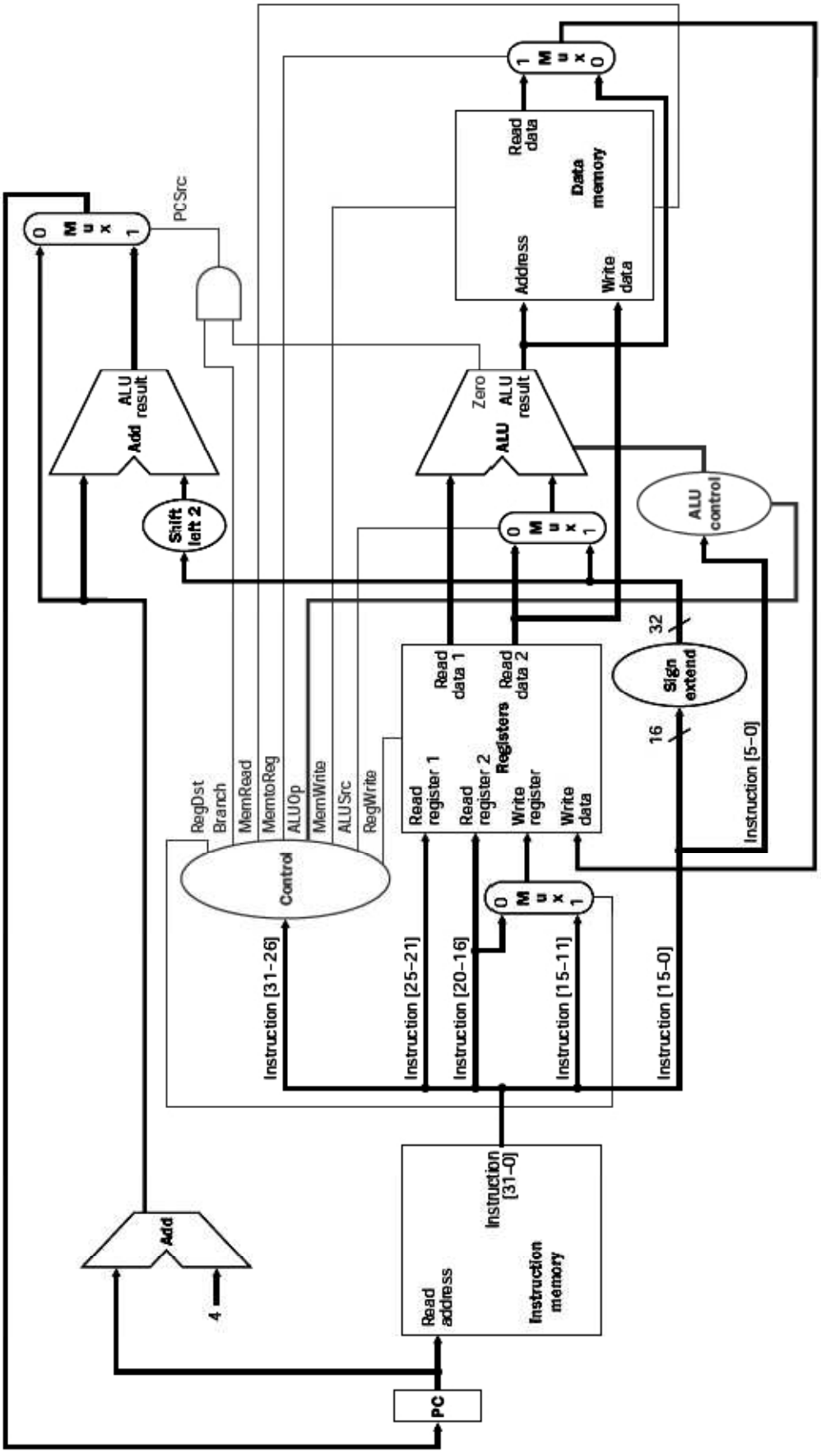


Abbildung 1 MIPS Einzyklendatenpfad

1.3) Der Befehlssatz soll jetzt um die Operation „shw r1“ (SwapHalfWord) in Form eines Pseudobefehls erweitert werden. Dabei werden die beiden Halbwörter eines 32-Bit Wortes vertauscht. Das folgende Beispiel verdeutlicht die Operation:

```
# $t0 = 0x1234ABCD
shw  $t0
# $t0 = 0xABCD1234
```

Implementieren Sie diesen Pseudobefehl und kommentieren Sie jede Zeile. **(1,5 Punkte)**

Hinweis: Nutzen Sie das Register $at (assembler-temporary), dass vom Assembler als temporäres Hilfsregister genutzt werden kann.

Aufgabe 4) Assembler (7 Punkte)

In der folgenden Aufgabe soll eine MIPS Assemblerfunktion geschrieben werden, die einzelne Buchstaben aus einem Array ausliest und alle enthaltenen Großbuchstaben in Kleinbuchstaben umwandelt. Die Buchstaben selbst sind als 32-Bit Werte im ASCII-Format kodiert. Als Hilfestellung sei folgende ASCII-Tabelle gegeben.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Schreiben Sie die Funktion „toLowercase“ in MIPS Assembler und halten Sie dabei alle MIPS Konventionen für Unterprogrammaufrufe ein. Die Basisadresse des Buchstabenarrays wird über das Parameterregister $a0 und die Länge des Arrays über das Parameterregister $a1 übergeben. Zusätzlich soll die Funktion die Anzahl der Umwandlungen zurückgeben. Sie können davon ausgehen, dass mindestens 1 Element in dem Array enthalten ist. Kommentieren Sie jede Zeile.

Auszug MIPS Befehlsreferenz

Tabelle 1

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 $s1,$s2,100	$s1 = $s2 - 100
add immediate unsigned	addiu $s1,$s2,100	$s1 = $s2 + 100
move from coprocessor register	mfc0 $s1, $epc	$s1 = $epc
multiply	mult $s2, $s3	Hi, Lo = $s2 x $s3
multiply unsigned	multu $s2, $s3	Hi, Lo = $s2 x $s3
divide	div $s2, $s3	Lo = $s2 : $s3, Hi = $s2 % $s3
divide unsigned	divu $s2, $s3	Lo = $s2 : $s3, Hi = $s2 % $s3
move from Hi	mfhi $s1	$s1 = Hi
move from Lo	mflo $s1	$s1 = Lo
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 2 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 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 $s1,$s2,10
bne	I	5	17	18	25			bne $s1,$s2,10
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

Tabelle 5 MIPS ALU Steuervektoren

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	010
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