Klausur Mikroprozessortechnik
24. Februar 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!

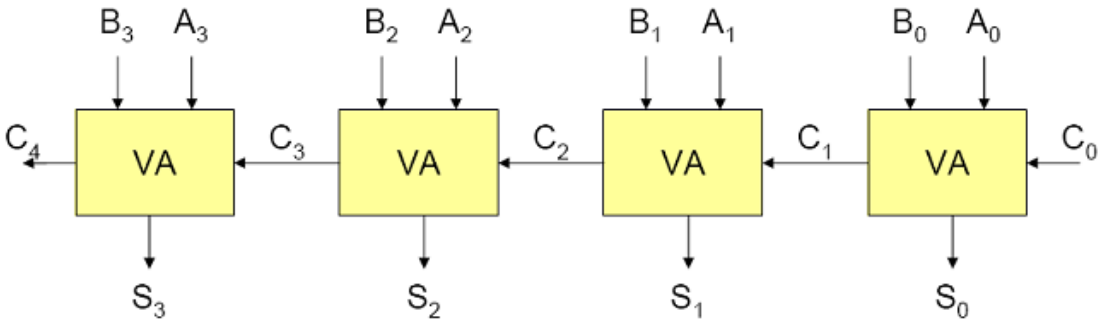
AUFGABE	PUNKTE
1	
2	
3	
4	
Gesamtpunkte	
Note	

Aufgabe 1) Zahlendarstellung (3,5 Punkte)

1.1) Wie viele Bits sind nötig, um die Zahl -334 im Zweierkomplement darzustellen? **(1 Punkt)**

1.2) Stellen Sie die Zahl -18.875 im 32-Bit Single-Precision-IEEE-754-Format dar! **(1,5 Punkte)**

1.3) Gegeben sei das Blockschaltbild eines Carry-Ripple-Addierers für 4-Bit Zahlen im Zweierkomplement. Fügen Sie eine entsprechende Logik für die Erkennung von Überläufen ein! **(1 Punkt)**



Aufgabe 2) Schaltwerksentwurf (6,5 Punkte)

In dieser Aufgabe soll ein binärer 2-Bit Zähler entworfen werden, der in Abhängigkeit von den beiden Eingangssignalen „up“ und „down“ **periodisch** von 0 bis 3 hoch- und runterzählen kann. (Der Nachfolger von der „Drei“ ist beim Hochzählen also wieder die „Null“ und der Nachfolger von der „Null“ ist beim Herunterzählen wieder die „Drei“) Wenn das Signal „up“ aktiv ist, soll hochgezählt, wenn das Signal „down“ aktiv ist, soll runtergezählt werden. Haben beide Eingangssignale den gleichen Pegel, soll der Zähler anhalten.

2.1) Entwerfen Sie ein geeignetes Zustandsdiagram! **(1 Punkt)**

2.2) Bestimmen Sie die minimalen Übergangsfunktionen! **(1,5 Punkte)**

2.3) Das Schaltwerk soll mit SR-FlipFlops realisiert werden! Bestimmen Sie die entsprechenden Beschaltungsfunktionen**(2 Punkte)**

2.4) Sie stellen nun fest, dass Ihnen ausschließlich D-FlipFlops zur Verfügung stehen. Wie können Sie ein SR-FlipFlop mithilfe eines D-FlipFlops umsetzen? Zeichnen Sie die Schaltung! **(1 Punkt)**

2.5) Was ist der Unterschied zwischen einem taktzustandsgesteuerten- und einem taktflankengesteuerten FlipFlop? **(1 Punkt)**

Aufgabe 3) Datenpfadentwurf (10 Punkte)

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

3.1) Nennen Sie die 5 MIPS-Adressierungsarten und geben sie jeweils einen passenden Befehl als Beispiel an. **(2.5 Punkte)**

3.2) 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
```

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

b) Untersuchen Sie nun die Daten- und Adressleitungen bei der Befehlsabarbeitung des Verzweigungsbefehls von Adresse 0x408 im Datenpfad. Füllen Sie dazu die einzelnen Felder in der Datenpfadabbildung 2.1 auf der nächsten Seite mit den richtigen Zahlenwerten aus. Sie können die Werte in binärer, dezimaler oder hexadezimaler Form angeben. **(3,5 Punkte)**

c) 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)**

d) Geben Sie die generierten Steuersignale für den betrachteten Verzweigungsbefehl in binärer Form an. Berücksichtigen Sie explizit Don't Care Fälle! **(2 Punkte)**

Steuersignal	Wert
RegDst	
Branch	
MemRead	
MemtoReg	
ALUOp	
ALUSrc	
RegWrite	
ALU Steuervektor	

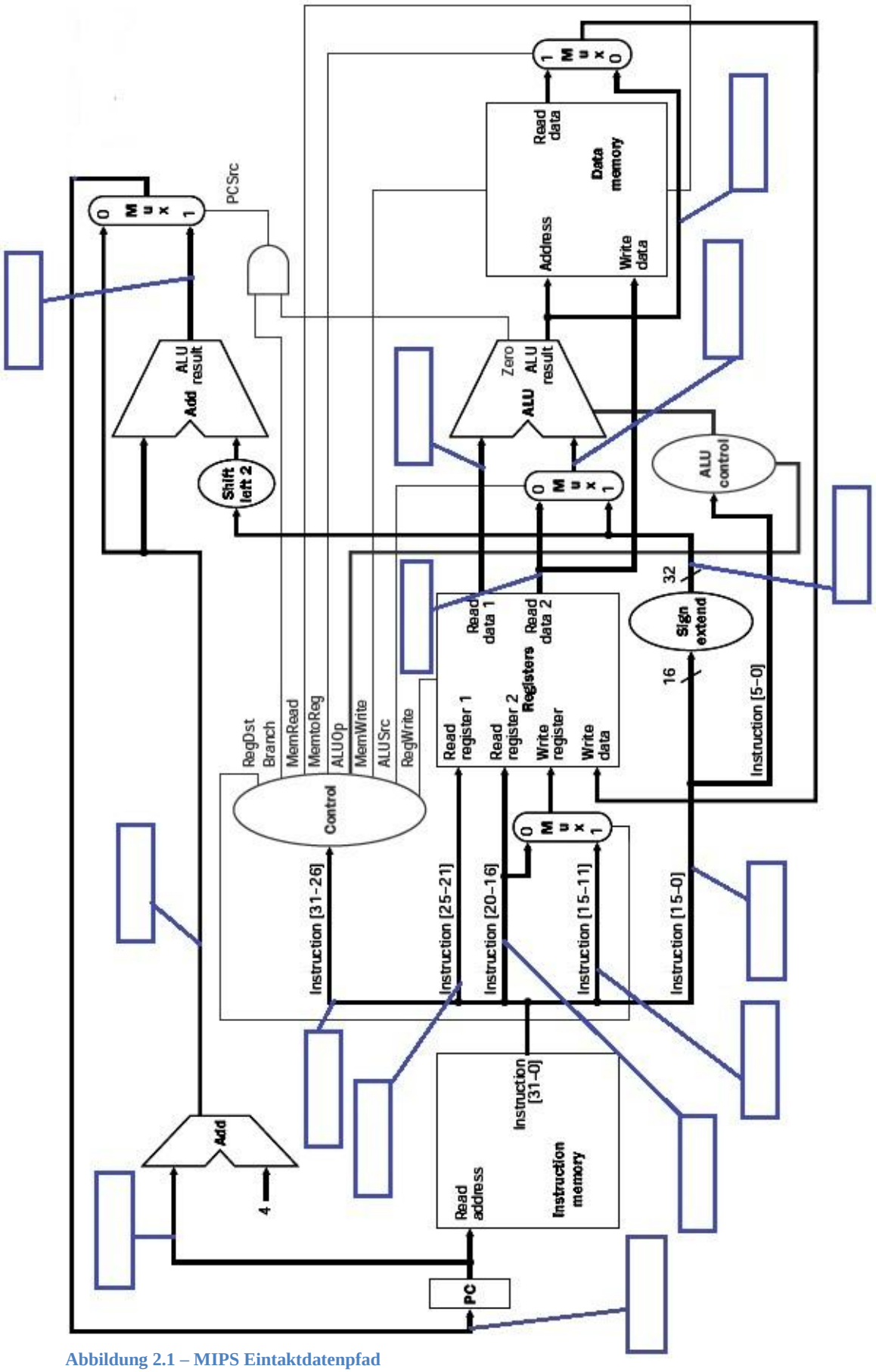


Abbildung 2.1 – MIPS Eintaktdatenpfad

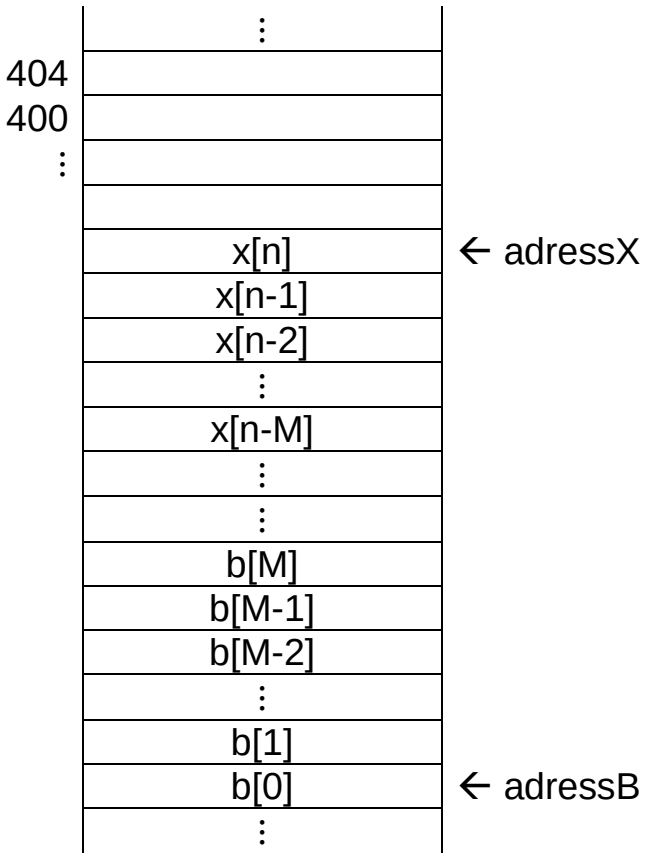
Aufgabe 4) Assembler (10 Punkte)

In der digitalen Signalverarbeitung können Filter mithilfe von Differenzengleichungen umgesetzt werden. Die folgende Gleichung zeigt die Differenzengleichung für ein einfaches FIR Filter der Ordnung M mit den (M+1)-Koeffizienten $b[k]$. Die Eingangswerte liegen in einem Array $x[n]$ und die entsprechenden Ausgangswerte werden in einem Array $y[n]$ gespeichert.

$$y[n] = \sum_{k=0}^{M} b[k] \cdot x[n-k]$$

In der folgenden Aufgabe soll nun eine Funktion in MIPS Assembler zur Berechnung eines beliebigen Ausgangswertes $y[n]$ implementiert werden.

Gehen Sie davon aus, dass die Koeffizienten $b[k]$ und die Eingangsfolge $x[n]$ wie folgt im Speicher abgelegt sind. Bei allen Werten handelt es sich um **32-Bit Zahlen im Zweierkomplement**.



4.1) In diesem Beispiel soll davon ausgegangen werden, dass der Prozessor über **keine** Multiplikationsbefehle verfügt! Implementieren Sie für diesen Zweck eine Funktion *„_multiplication“* in MIPS Assembler, die zwei **32 Bit** Zahlen multipliziert und das Produkt als **32-Bit** Zahl zurückgibt. Eventuell auftretende Überläufe sollen **ignoriert** werden. Sie können außerdem davon ausgehen, dass ausschließlich mit **positiven Zahlen** gearbeitet wird. Berücksichtigen Sie alle Konventionen für MIPS Unterprogrammaufrufe! **(2 Punkte)**
Hinweis: $5 \cdot 3 = 3 + 3 + 3 + 3 + 3$

4.2) Implementieren Sie nun die Funktion *„_firCalculation“* zur Berechnung eines Ausgangswertes y[n]! Die Funktion benötigt **drei Parameter**:

- Parameter 1 beinhaltet die Adresse des *n*-ten Eingangswerts $x[n]$ wie im Speicherabbild skizziert.
- Parameter 2 beinhaltet die Adresse des ersten Koeffizienten $b[0]$ ebenfalls wie im Speicherabbild skizziert.
- Parameter 3 beinhaltet die Filterordnung M (stets eine positive Zahl größer als Null).

Die Funktion soll den berechneten Ausgangswert y[n] zurückgeben. Kommentieren Sie jede Zeile! Verwenden Sie für die Multiplikationsoperation Ihre Hilfsfunktion aus der vorhergehenden Teilaufgabe! (**8 Punkte)**

Hinweis:
Beachten Sie alle MIPS Unterprogrammaufrufkonventionen. Bedenken Sie auch, dass hier eine Funktion in einer Funktion aufgerufen wird! Treffen Sie alle nötigen Vorkehrungen.

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	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

Tabelle 5 MIPS – ALU Steuervektoren