Klausur Mikroprozessortechnik
21. Februar 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 vollständig erkennbar ist.
- Die Bearbeitungszeit beträgt **120** Minuten.

Viel Erfolg!

AUFGABE	PUNKTE
1	
2	
3	
4	
Gesamtpunkte	
Note	

Aufgabe 1) Zahlendarstellungen (6 Punkte)

a) Wie viele Bits sind nötig, um die Zahl -231 im Zweierkomplement dar zustellen? **(1 Punkt)**

b) Stellen Sie die Zahl 14.625 im 32-Bit Single-Precision-IEEE-754-Format dar! **(1,5 Punkte)**

c) Berechnen Sie folgende Addition im 8-Bit Zweierkomplement und beurteilen Sie das Ergebnis! Setzen Sie die entsprechenden Statusbits. **(3,5 Punkte)**

-7 + 49

Statusbits:

Overflow	Carry	Sign	Zero

Aufgabe 2) Schaltwerksentwurf (9 Punkte)

Gegeben sei das folgende Zustandsdiagramm, mit dem Eingang *e* und dem Ausgang *y*, welcher nur in den gestreiften Zuständen den Wert 1 annimmt.

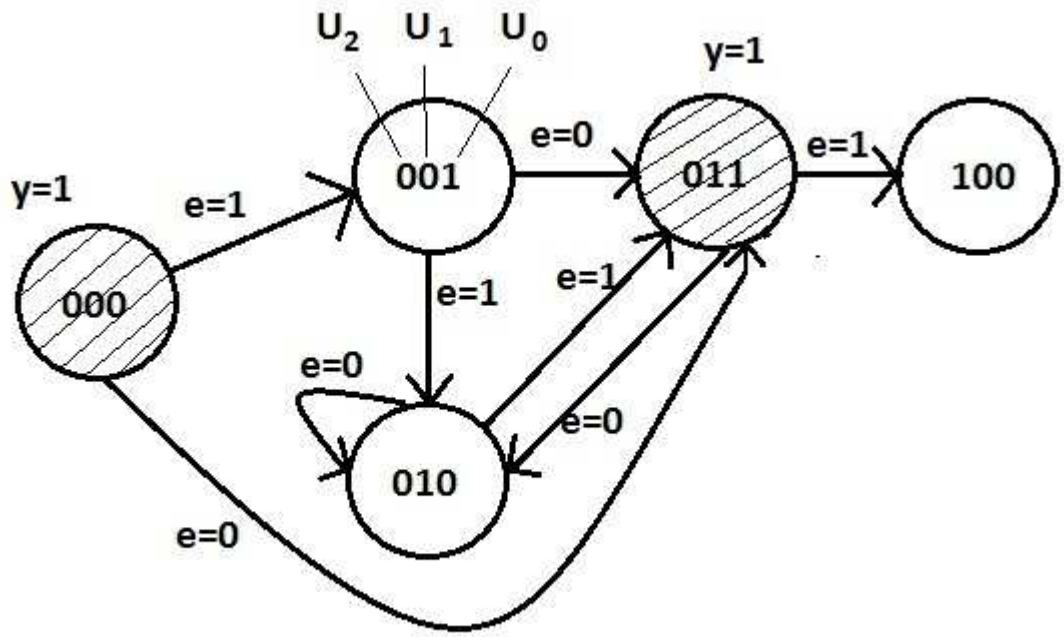

Abbildung 1 – Zustandsdiagramm

a) Handelt es sich hierbei um einen Mealy- oder einen Moore-Automaten? Begründen Sie kurz. **(1 Punkt)**

b) Bestimmen Sie die minimalen Übergangsfunktionen! **(3 Punkte)**

c) Bestimmen Sie die Beschaltungsfunktion für ein SR-FlipFlop, dass das Bit für U_1 speichern soll. **(1 Punkt)**

d) Betrachten Sie wieder das gleiche Zustandsdiagramm und die 2 folgenden Impulsdiagramme. Entscheiden Sie für jedes Impulsdiagramm, ob es zu dem gegebenen System passt oder nicht. Setzen Sie für den zutreffenden Fall die Signalverläufe von s_1, r_1, d_0 *und* e fort oder markieren Sie im anderen Fall die entsprechende Diskrepanz. Gatterlaufzeiten wurden willkürlich gewählt und brauchen nicht berücksichtigt werden.

(2 Punkte)

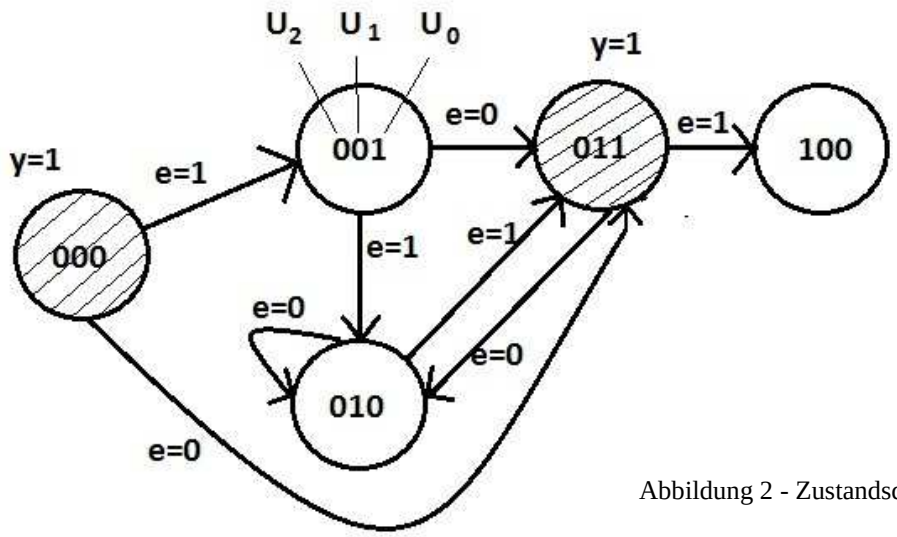

Abbildung 2 - Zustandsdiagramm

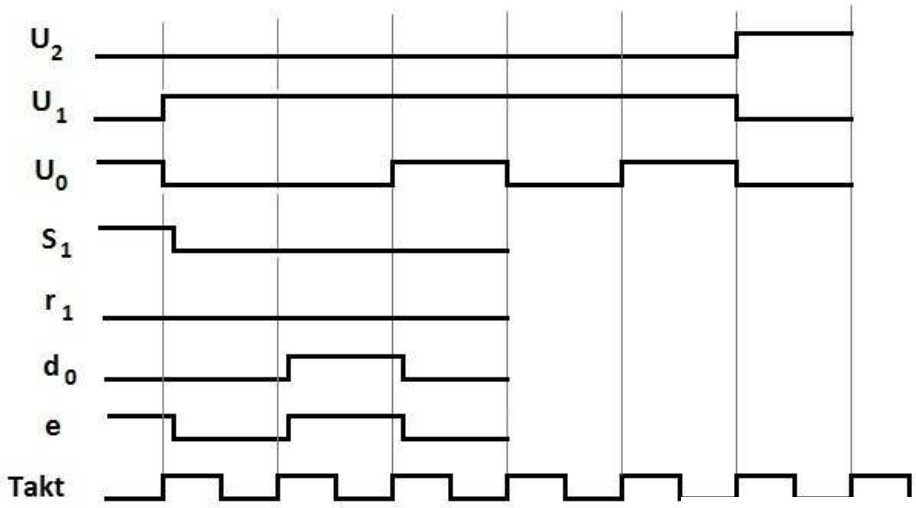

Abbildung 3 – Impulsdiagramm a)

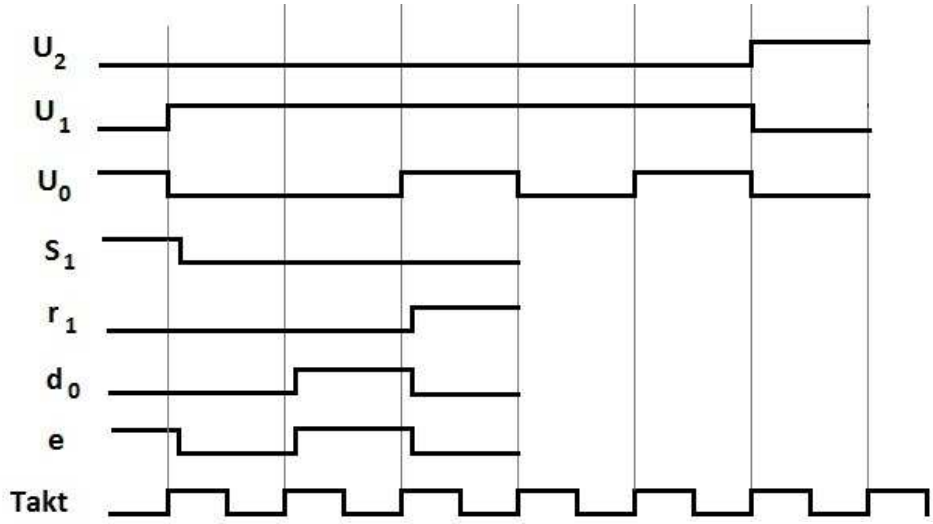

Abbildung 4 – Impulsdiagramm b)

e) Begründen Sie kurz, ob hier asynchrone, zustandsgesteuerte, einflankengesteuerte oder zweiflankengesteuerte FlipFlops verwendet wurden. **(1 Punkt)**

f) Bestimmen Sie die minimale disjunktive Normalform für das Ausgangssignal y, welches nur in den gestreiften Zuständen den Wert „1“ annimmt. **(1 Punkt)**

Aufgabe 3) Datenpfadentwurf (8 Punkte)

Beziehen Sie sich zunächst den MIPS Pipelinedatenpfad wie in Abbildung 5 dargestellt.

a) Handelt es sich bei den folgenden Elementen im Pipelinedatenpfad um ein Schaltnetz oder ein Schaltwerk? (Falsche Antworten führen zu Punktabzug, weniger als 0 Punkte sind nicht möglich, Begründungen sind nicht gefordert) **(3 Punkte)**

Konflikterkennung
Steuerwerk
Registerbank
MEM/WB Puffer
Multiplexer vor den ALU Eingängen
Bypassing-Einheit

b) Während der Ausführung im Pipelinedatenpfad wird ausschließlich der folgende Konflikt in der Forwardingeinheit detektiert:

(MEM/WB.RegWrite) and (MEM/WB.RegisterRd != 0) and
(EX/MEM.RegisterRd != ID/EX.RegisterRt) and
(MEM/WB.RegisterRd = ID/EX.RegisterRt) **(2 Punkte)**

Um welche Art von Konflikttyp handelt es sich hierbei?

Konstruieren Sie ein kurzes Assemblerbeispiel, wo genau dieser Fall auftritt!

Wie müssen die Multiplexer am Eingang der ALU in diesem Fall angesteuert werden?

ForwardA= ForwardB=

c) Gehen Sie nun davon aus, dass die Forwarding-Einheit und die Einheit zum Erkennen von Konflikten **nicht** im Datenpfad integriert sind. Gehen Sie weiter davon aus, dass die Registerbank in einem Takt geschrieben und gelesen werden kann, ohne dass ein Datenkonflikt auftritt. Beziehen Sie sich auf folgendes Codefragment:

```
loop:  lw    $s5,0($s4)
       lw    $t3,40($s4)
       add   $s4,$t3,$t3
       ori   $t3,$sp,0x20
       lw    $s5,20($s3)
       sll   $s5,$t0,2
       sll   $s4,$s3,5
       xor   $s5,$t4,$s5
       and   $t7,$t8,$t4
       j     loop
```

Fügen Sie zunächst ausreichend NOP Befehle ein, damit die Schleife konfliktfrei durchlaufen wird. **(1,5 Punkte)**

d) Minimieren Sie die durch geschicktes Umstellen der Befehlsabfolge die Anzahl der benötigten Takte. **(1,5 Punkte)**

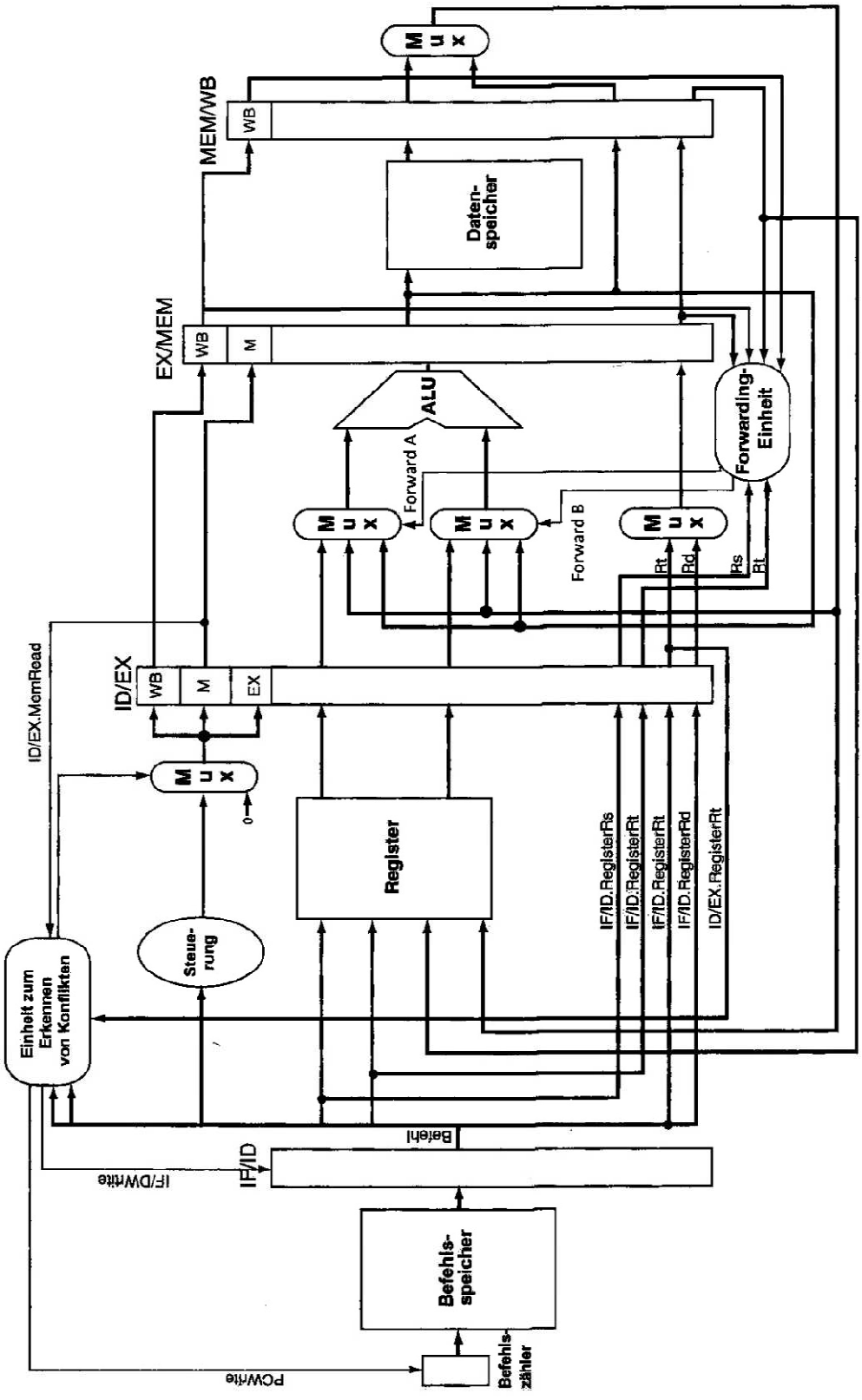

Abbildung 5 – MIPS Pipelinedatenpfad

Aufgabe 4) Assembler (7 Punkte)

a) Schreiben Sie eine Funktion „maxDistance“ in MIPS Assembler, die für ein Integer-Array mit positiven Werten den größten Abstand benachbarter Zahlen berechnet und den Index der ersten Zahl zurückgibt (Bei mehrfachen Vorkommen den ersten Index). Das erste Arrayelement hat den Index 0. Zum Verständnis sind Ihnen zwei Beispiele gegeben:

Beispiel 1:
array1: 12 5 9 20 2 8 7

Größter Abstand zwischen dem 4. Element (20) und 5. Element (2)
Index der Zahl 20: 3 → Rückgabewert: 3

Beispiel 2:
array1: 8 5 21 6 2 8 7

Größter Abstand zwischen dem 2. Element (5) und 3. Element (21)
Index der Zahl 5: 1 → Rückgabewert: 1

Im Register $a0 befindet sich die Basisadresse des Arrays. Im Register $a1 befindet sich die Länge des Arrays. Halten Sie alle Konventionen für MIPS Unterprogrammaufrufe ein! Kommentieren Sie jede Zeile.

maxDistance:

Auszug 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 $s1,$s2,100	$s1 = $s2 - $s3
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

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	$d = u^+$	$s = u^+\|_{u=0}$ $r = \overline{u^+}\|_{u=1}$	$j = u^+\|_{u=0}$ $k = \overline{u^+}\|_{u=1}$	$t = u^+ \oplus u$