Mikroprozessortechnik WS 09/10
Probeklausur

29. Januar 2010

1. **<u>Gleitkommaarithmetik nach IEEE-754 (5 Punkte)</u>**

Die folgenden Umwandlungen/Berechnungen beziehen sich auf das 32-Bit Single-Precision Format nach IEEE-754.

Stellen Sie die Zahl 7,625 in folgender Tabelle dar! **(1 Punkt)**

Führen Sie für die gegebenen Zahlen eine Addition durch.**(2 Punkte)**

1	1	0	0	0	0	0	1	1	0	0	1	0	1	1	0	0	1	1	1	0	1	1	0	0	0	1	0	0	0	0	0
0	0	1	1	1	1	1	1	0	1	0	0	1	1	1	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0	0	0

Wie werden denormalisierte Zahlen abgespeichert? **(1 Punkt)**

Was ist die kleinste darstellbare denormalisierte Zahl? **(1 Punkt)**

2. Schaltnetzentwurf (4 Punkte)

Füllen Sie die gegebene KV-Tafel für die gegebene Funktion **(1 Punkt)**

Y = $\bar{a}d + \overline{\bar{c} + d} + (\bar{a} + \bar{b})\bar{c}$

Realisieren Sie eine CMOS-Transistorschaltung für folgende Funktion **(3 Punkte)**
w = $\bar{c} + ab + \overline{ba}$

3. Schaltwerksentwurf (7 Punkte)

Gegeben sei eine 1-Bit Leitung, über die Daten seriell übertragen werden können.
Erkennen Sie in diesem seriellen Datenstrom die Bitmuster „001“, „11“ und „10“! Sobald ein Bitmuster erkannt wurde soll ein Ausgangssignal z für einen Takt lang aktiv sein.

a) Entwerfen Sie ein Zustandsdiagramm mit passender Codierung. **(1 Punkt)**

b) Bestimmen Sie die Übergangsfunktionen. **(3 Punkte)**

c) Bestimmen Sie die Beschaltungsfunktionen für die Flip-Flops. Verwenden Sie mindestens ein SR-FlipFlop, ein JK-FlipFlop sowie ein D-FlipFlop! **(3 Punkte)**

d) Bestimmen Sie die konjunktive Normalform für das Ausgangssignal z! **(1 Punkt)**

4. Datenpfad (7 Punkte)

Betrachten Sie vorerst den aus der Vorlesung bekannten Mehrzykendatenpfad.

a) Beschreiben Sie die Funktionen der Steuersignale RegDst sowie ALUOp. **(1 Punkt)**

b) Der Datenpfad soll an dieser Stelle um einen neuen Befehl „superAdd r0,r1,r2“ erweitert werden. Es gilt:

superAdd r0,r1,r2 r0 = 2*(r1 + r2)

Erweitern Sie den Datenpfad zunächst um zusätzliche Elemente, um die Ausführung dieses Befehls möglich zu machen. Nutzen Sie dazu die Datenpfadabbildung im Anhang **(1 Punkt)**

c) Beschreiben Sie nun die Abarbeitung des Befehls, in dem Sie für alle Stufen die relevanten Steuersignale erzeugen! **(2 Punkte)**

Beziehen Sie sich für die folgende Aufgabe auf den MIPS Pipelinedatenpfad aus der Vorlesung ohne Forwarding und ohne Stalling!

a) Gegeben sei folgendes Codefragment

```
        sw      $t1,4($sp)
        add     $s0,$s1,$s2
        beq     $s0,$s1,Exit
        nor     $t5,$sp,$sp
        lw      $s3,0($sp)
        sll     $s2,$s3,5
Exit:
```

Welche Konflikte treten hier auf, wie werden sie genannt? **(1 Punkt)**

b) Lösen Sie die Konflikte indem Sie den Code umstellen und möglichst wenige NOP's einfügen. **(1 Punkt)**

c) Wie funktioniert ein 2-Bit Sprungvorhersagepuffer, wie kann man ihn realisieren? Zeichnen Sie das Zustandsdiagramm. **(1 Punkt)**

5. MIPS Assembler (7 Punkte)

Schreiben Sie eine Funktion *convertByteArray(int source, int destination,int elements)* in MIPS Assembler, die startend an der Quelladresse eine Anzahl von Wörtern aus dem Speicher liest. In einem Wort sind 4 Bytes im Little-Endian Format codiert. Die einzelnen Bytes eines Wortes sollen nun in den Speicher startend an der Zieladresse zurückkopiert werden. Jedes Byte an eine Wortspeicherzelle im Speicher.

a) Gegeben sei der Speicher vor dem Aufruf der Funktion *convertByteArray(0x200,0x220,3)* Tragen Sie die Werte der Speicherelemente nach dem Aufruf der Funktion ein! **(2 Punkte)**

0x250		0x250	
0x24C		0x24C	
0x248		0x248	
0x244		0x244	
0x240		0x240	
0x23C		0x23C	
0x238		0x238	
0x234		0x234	
0x230		0x230	
0x22C		0x22C	
0x228		0x228	
0x224		0x224	
0x220		0x220	
0x21C		0x21C	
0x218		0x218	
0x214		0x214	
0x210		0x210	
0x20C		0x20C	
0x208	0x656C6767	0x208	
0x204	0x754A634D	0x204	
0x200	0x6B69614D	0x200	
0x19C		0x19C	

b) Betrachten Sie nun den gegebenen Coderumpf auf der nächsten Seite. Beschreiben Sie in 2 Sätzen, was in den Zeilen 16 bis 25 berechnet wird. Kommentieren Sie den Code. **(2 Punkt)**

c) Implementieren Sie die Funktionalität der Methode in den darauf folgenden Zeilen **(3 Punkte)**

```
1: #    MIPS Assembler
2: #    void convertByteArray(int source,int destination,int elements)
3:
4: .data
5: array: .word 0x4D 0x61 0x69 0x6B 0x4D 0x63 0x4A 0x75 0x67 0x67 0x6C 0x65
6:
7: .text
8: .globl main
9: main:                la      $a0,array
10:                     addi    $a1,$a0,0x20
11:                     li      $a2,3
12:                     jal     convertByteArray
13:                     li      $v0,4
14:                     syscall
15:
16: convertByteArray:   addi    $sp,$sp,-8          # 2 elements on stack
17:                     sw      $s0,4($sp)
18:                     sw      $s1,0($sp)
19:                     sll     $s1,$a2,2           # byte address
20:                     add     $s0,$a0,$s1         # source address + offset
21:                     bgt     $a1,$s0,label_1
22:                     sll     $s1,$s1,2
23:                     sub     $s0,$a0,$s1         # source address - offset
24:                     ble     $a1,$s0,label_1
25:                     j       Exit
26: label_1:
27:
28:
29:
30:
31:
32:
33:
34:
35:
36:
37:
38:
39:
40:
41:
42:
43:
44:
45:
46:
47:
48: Exit:
49:
50:
51:
```

MIPS Registerübericht und Wertetabelle der ALUSteuerungseinheit.

Name	Register Number	Usage	Preserved on call
$zero	0	the constant value 0	n.a.
$at	1	reserved for the assembler	n.a.
$v0-$v1	2-3	value for results and expressions	no
$a0-$a3	4-7	arguments (procedures/functions)	yes
$t0-$t7	8-15	temporaries	no
$s0-$s7	16-23	saved	yes
$t8-$t9	24-25	more temporaries	no
$k0-$k1	26-27	reserved for the operating system	n.a.
$gp	28	global pointer	yes
$sp	29	stack pointer	yes
$fp	30	frame pointer	yes
$ra	31	return address	yes

Opcode	ALUOp	Operation	funct-field	ALU Operation	ALU ControlInput
LW	00	load word	******	Addition	0010
SW	00	store word	******	Addition	0010
Branch on equal	01	branch on 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

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
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
set less than immediate unsigned	sltiu $s1,$s2,100	if($s2<100) $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	$u^+ = \mathrm{d}$	$u^+ = \bar{u}s + u\bar{r}$	$u^+ = j\bar{u} + \bar{k}u$	$u^+ = \bar{t}u + t\bar{u}$
Beschaltungsfunktion	$d = u^+$	$s = u^+\|_{u=0}$ $r = \overline{u^+}\|_{u=1}$	$j = u^+\|_{u=0}$ $k = \overline{u^+}\|_{u=1}$	$\mathrm{t} = u^+ \oplus u$