

Klausur zu
Technische Grundlagen der Informatik 2
Rechnerorganisation
(TechGI2 und TechGI2TI)

26. September 2011

Fakultät IV
Institut für Technische Informatik
und Mikroelektronik
FG Architektur eingebetteter Systeme

Nachname :
Vorname :
Matrikelnummer :
Studiengang :	<input type="radio"/> Inf <input type="radio"/> TI <input type="radio"/> Wi-Ing <input type="radio"/> Andere:
Prüfungsordnung :	<input type="radio"/> Bachelor <input type="radio"/> Diplom
Prüfungsform :	<input type="radio"/> Schriftliche Prüfung <input type="radio"/> Prüfungsäquivalente Studienleistungen

Aufgabe	1	2	3	4	5	Σ
max. Punkte	9	7	9	4	6	35
erreichte Punkte						
Korrektor						

Wichtige Hinweise:

- Deckblatt bzgl. der Personenangaben ausfüllen.
- Kopf aller abgegebenen Seiten ausfüllen, d.h. mit Namen und Matrikelnummer versehen.
- Die Klausur dauert 75 Minuten.
- Bei jeder Aufgabe steht, wieviele Punkte erzielt werden können.
- Für die Lösungen sind die Aufgabenblätter zu verwenden; Zusatzblätter werden nach Bedarf ausgegeben.
- Es dürfen weder Bleistift noch Rotstift verwendet werden.
- Der Lösungsweg muss nachvollziehbar sein.
- Das Endergebnis muss deutlich gekennzeichnet sein.
- Ein nicht-programmierbarer Taschenrechner darf verwendet werden.

- Mobiltelefone sind auszuschalten.
- Betrugsversuche werden mit einem Nichtbestehen des Moduls bzw. der Klausur geahndet.

1. Aufgabe: MIPS Assemblersprache (9 Punkte)

Übersetzen Sie die folgende C-Funktion nach MIPS-Assembler. Beachten Sie dabei die MIPS-Registerkonventionen. Pseudobefehle dürfen verwendet werden.

```
int count_between(int a[], int n, int min, int max)
{
    int i, count;

    count = 0;
    for (i=0; i<n; i++)
        if (min=>a[i] && a[i]<=max)
            count++;

    return count;
}
```

2. Aufgabe: Zahlendarstellung (7 Punkte)

- a) Welche Dezimalzahl stellt die folgende Zweierkomplementzahl dar?

1111 1111 1111 1111 1111 1111 1101 1111

- b) Folgende Gleitkommazahlen sind nach IEEE 754 mit einfacher Genauigkeit dargestellt:

P = 1111 1000 0100 1111 1111 1111 1111 1111

Q = 1111 1000 0101 0111 1111 1111 1111 1111

R = 0000 0111 1100 1111 1111 1111 1111 1111

S = 0111 1000 0101 0111 1111 1111 1111 1111

Ordnen Sie diese Zahlen von klein nach groß. Begründen Sie Ihre Antwort. Der Lösungsweg muss klar erkennbar sein.

3. Aufgabe: Eintaktprozessor (9 Punkte)

In der Abbildung auf der folgenden Seite ist der Datenpfad des Eintaktprozessors erweitert worden, um den `multiply` (MULT) Befehl zu implementieren. Eine Multiplikationseinheit wurde hinzugefügt, sowie die *Hi*- und *Lo*-Register, die geschrieben werden, wenn das *HiLoWrite*-Steuersignal gesetzt ist. Um den MULT-Befehl nutzen zu können, werden auch die Befehle `move from hi` (MFHI) und `move from lo` (MFLO) benötigt. Das Format und die Bedeutung dieser Befehle sind:

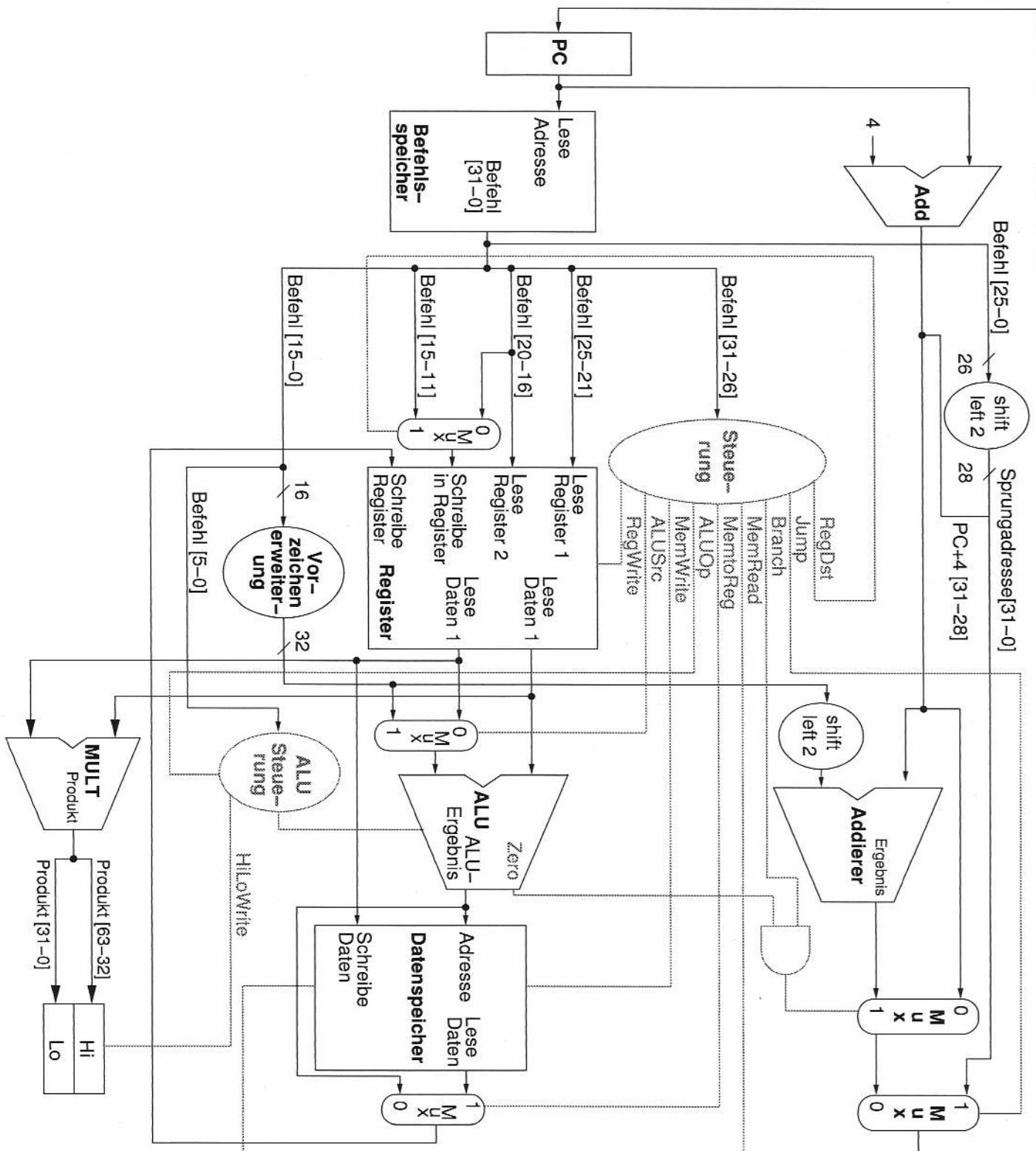
	Opcode	rs	rt	rd	shamt	funct	
MFHI <i>rd</i>	0	0	0	rd	0	0x10	Kopiere Inhalt des Registers <i>Hi</i> nach Allzweckregister <i>rd</i> .
MFLO <i>rd</i>	0	0	0	rd	0	0x12	... <i>Lo</i> nach Allzweckregister <i>rd</i> .

Aufgaben:

- Tragen Sie in die gegebene Zeichnung ein, wie der Datenpfad des Eintaktprozessors geändert werden muss, um die MFHI- und MFLO-Befehle zu implementieren. Erläutern Sie kurz die angedachte Funktionsweise.
- Geben Sie an, welche Werte die Steuersignale haben müssen, sodass der Datenpfad die MFHI- und MFLO-Befehle ausführt. Benutzen Sie *don't cares* wenn möglich.

zu b)

Signal	Wert (MFHI/MFLO)
RegDst:	
Jump:	
Branch:	
MemRead:	
MemToReg:	
ALUOp:	
MemWrite:	
ALUSrc:	
RegWrite:	
HiLoWrite:	

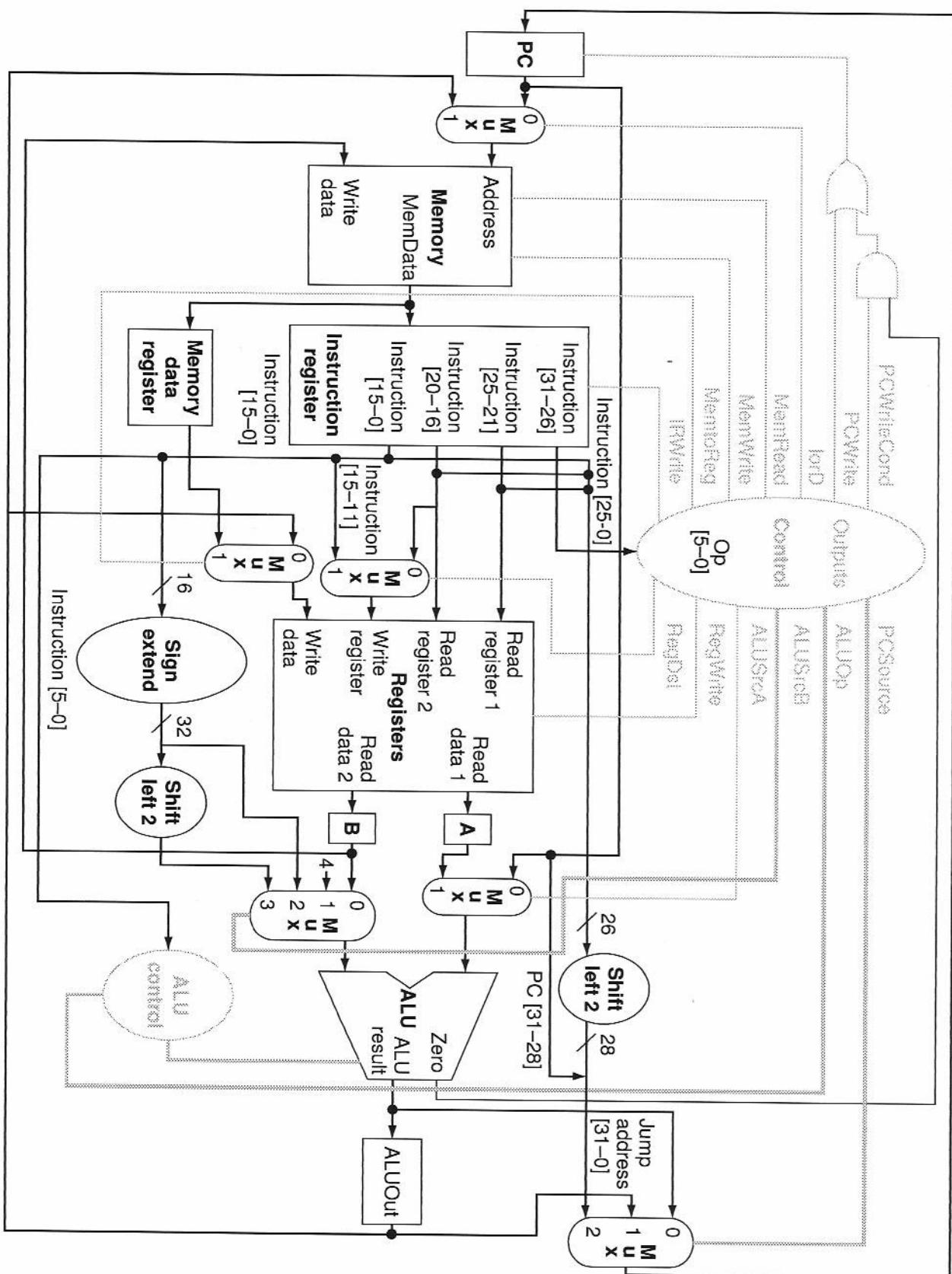


4. Aufgabe: Mehrzyklen- und Pipelineprozessor (4 Punkte)

- a) Die Abbildung auf der folgenden Seite zeigt die Mehrzyklenimplementierung. Geben Sie an, welche Werte die Steuersignale haben müssen, damit der Datenpfad den ersten Schritt (den Befehlsholschritt) ausführt. Benutzen Sie *don't cares* wenn möglich.

Signal	Wert
ALUOp:	
ALUSrcA:	
ALUSrcB:	
IRWrite:	
IorD:	
MemRead:	
MemWrite:	
MemtoReg:	
PCSource:	
PCWrite:	
PCWriteCond:	
RegDst:	
RegWrite:	

- b) Warum ist der Entwurf des Pipeline-Datenpfades durch die Tatsache, dass alle MIPS-Befehle gleich lang sind, vereinfacht?



5. Aufgabe: Caches (6 Punkte)

- a) Ein 4-fach satzassoziativer Cache ist 32 KB groß und verwendet Blöcke von 32 Bytes. Die Adresslänge ist 32 Bit und der Speicher ist Byte-adressierbar.

Wieviele Bits werden jeweils für *Offset*, *Index* und *Tag* benötigt?

- b) Der Prozessor erzeugt folgende Adresse (als Hexadezimalzahl): $9\text{bdf } 7531_{\text{H}}$.

- Auf welchen *Satz* wird diese Adresse abgebildet und was ist der *Tag*?
- Geben Sie den *Index* als Dezimalzahl und den *Tag* als Hexadezimal- oder Dualzahl an.

MIPS Reference Data Card ("Green Card") 1. Pull along perforation to separate card 2. Fold bottom side (columns 3 and 4) together

MIPS Reference Data

(1)


ARITHMETIC CORE INSTRUCTION SET

(2)

NAME, MNEMONIC	FOR-MAT	OPERATION	OPCODE / FMT / FT / FUNCT (Hex)
Branch On FP True	bclt	FI if(FPcond)PC=PC+4+BranchAddr (4)	11/8/1/-
Branch On FP False	bclf	FI if(!FPcond)PC=PC+4+BranchAddr(4)	11/8/0/-
Divide	div	R Lo=R[rs]/R[rt]; Hi=R[rs] % R[rt]	0/-/-/1a
Divide Unsigned	divu	R Lo=R[rs]/R[rt]; Hi=R[rs] % R[rt] (6)	0/-/-/1b
FP Add Single	add.s	FR F[fd]=F[fs] + F[ft]	11/10/-/0
FP Add	add.d	FR {F[fd],F[fd+1]} = {F[fs],F[fs+1]} + {F[ft],F[ft+1]}	11/11/-/0
Double	cxd*	FR FPcond = (F[fs] op F[ft]) ? 1 : 0	11/10/-/y
FP Compare Single	cxs*	FR FPcond = ((F[fs],F[fs+1]) op {F[ft],F[ft+1]}) ? 1 : 0	11/11/-/y
FP Compare	cxd*	* (x is eq, lt, or le) (op is ==, <, or <=) (y is 32, 3c, or 3e)	
Double	cxd*	FR FPcond = (F[fs],F[fs+1]) / {F[ft],F[ft+1]}	11/11/-/y
FP Divide Single	div.s	FR F[fd]=F[fs] / F[ft]	11/10/-/3
FP Divide	div.d	FR {F[fd],F[fd+1]} = {F[fs],F[fs+1]} / {F[ft],F[ft+1]}	11/11/-/3
FP Multiply Single	mul.s	FR F[fd]=F[fs] * F[ft]	11/10/-/2
FP Multiply	mul.d	FR {F[fd],F[fd+1]} = {F[fs],F[fs+1]} * {F[ft],F[ft+1]}	11/11/-/2
Double	mult	FR F[fd]=F[fs] - F[ft]	11/10/-/1
FP Subtract Single	sub.s	FR F[fd]=F[fs] - F[ft]	11/10/-/1
FP Subtract	sub.d	FR {F[fd],F[fd+1]} = {F[fs],F[fs+1]} - {F[ft],F[ft+1]}	11/11/-/1
Double	mult	I F[rt]=M[R[rs]+SignExtImm] (2)	31/-/-/-
Load FP Single	lwc1	I F[rt]=M[R[rs]+SignExtImm]; R[rt+1]=M[R[rs]+SignExtImm+4]	35/-/-/-
Load FP	ldc1	I R[rd]=Hi	0/-/-/10
Double	ldc1	R R[rd]=Lo	0/-/-/12
Move From Hi	mfhi	R R[rd]=CR[rs]	10/0/-/0
Move From Lo	mflo	R R[rd]=R[rt]	0/-/-/18
Move From Control	mfco	R R[rd]=CR[rs]	0/-/-/19
Multiply	mult	R {Hi,Lo} = R[rs] * R[rt]	0/-/-/18
Multiply Unsigned	multu	R {Hi,Lo} = R[rs] * R[rt] (6)	0/-/-/19
Shift Right Arith.	sra	R R[rd] = R[rt] >> shamt	0/-/-/3
Store FP Single	swc1	I M[R[rs]+SignExtImm] = F[rt] (2)	39/-/-/-
Store FP	sdc1	I M[R[rs]+SignExtImm] = F[rt]; M[R[rs]+SignExtImm+4] = F[rt+1]	3d/-/-/-

FLOATING-POINT INSTRUCTION FORMATS

FR	opcode	fmt	ft	fs	fd	funct	0
31	26 25	21 20	16 15	11 10	6 5	0	
FI	opcode	fmt	ft	immediate			0
31	26 25	21 20	16 15				

PSEUDOINSTRUCTION SET

NAME	MNEMONIC	OPERATION
Branch Less Than	blt	if(R[rs]<R[rt]) PC = Label
Branch Greater Than	bgt	if(R[rs]>R[rt]) PC = Label
Branch Less Than or Equal	ble	if(R[rs] <= R[rt]) PC = Label
Branch Greater Than or Equal	bge	if(R[rs] >= R[rt]) PC = Label
Load Immediate	li	R[rd] = immediate
Move	move	R[rd] = R[rs]

REGISTER NAME, NUMBER, USE, CALL CONVENTION

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
\$zero	0	The Constant Value 0	N.A.
\$at	1	Assembler Temporary	No
\$v0-\$v1	2-3	Values for Function Results and Expression Evaluation	No
\$a0-\$a3	4-7	Arguments	No
\$t0-\$t7	8-15	Temporaries	No
\$s0-\$s7	16-23	Saved Temporaries	Yes
\$t8-\$t9	24-25	Temporaries	No
\$k0-\$k1	26-27	Reserved for OS Kernel	No
\$gp	28	Global Pointer	Yes
\$sp	29	Stack Pointer	Yes
\$fp	30	Frame Pointer	Yes
\$ra	31	Return Address	Yes

BASIC INSTRUCTION FORMATS

R	opcode	rs	rt	rd	shamt	funct	0
	31	26 25	21 20	16 15	11 10	6 5	0
I	opcode	rs	rt			immediate	0
	31	26 25	21 20	16 15			0
J	opcode				address		0
	31	26 25					0

(1) May cause overflow exception
 (2) SignExtImm = { 16{immediate[15]}, immediate }
 (3) ZeroExtImm = { 16{1'b0}, immediate }
 (4) BranchAddr = { 14{immediate[15]}, immediate, 2'b0 }
 (5) JumpAddr = { PC+4[31:28], address, 2'b0 }
 (6) Operands considered unsigned numbers (vs. 2's comp.)
 (7) Atomic test&set pair; R[rt] = 1 if pair atomic, 0 if not atomic

OPCODES, BASE CONVERSION, ASCII SYMBOLS

MIPS (1) MIPS (2) MIPS	opcode funct funct	Binary	Deci-mal	Hexa-deci-mal	ASCII Char-acter	Deci-mal	Hexa-deci-mal	ASCII Char-acter
(31:26)	(5:0)							
(1)	sll addf	00 0000	0	0 NUL	64 40	@		
	subf	00 0001	1	1 SOH	65 41	A		
j	srl mulf	00 0010	2	2 STX	66 42	B		
jal	sra divf	00 0011	3	3 ETX	67 43	C		
beq	slv sqrf	00 0100	4	4 EOT	68 44	D		
bne	absf	00 0101	5	5 ENQ	69 45	E		
blez	sriv movf	00 0110	6	6 ACK	70 46	F		
bgtz	srav negf	00 0111	7	7 BEL	71 47	G		
addi	jr	00 1000	8	8 BS	72 48	H		
addiu	jalr	00 1001	9	9 HT	73 49	I		
slti	movz	00 1010	10	a LF	74 4a	J		
sltiu	movn	00 1011	11	b VT	75 4b	K		
andi	syscall roundwf	00 1100	12	c FF	76 4c	L		
ori	break truncwf	00 1101	13	d CR	77 4d	M		
xori	ceilwf	00 1110	14	e SO	78 4e	N		
lui	sync floorwf	00 1111	15	f SI	79 4f	O		
(2)	mfhi	01 0000	16	10 DLE	80 50	P		
mthi	01 0001	17	11 DC1	81 51	Q			
mflo	movzf	01 0010	18	12 DC2	82 52	R		
mtlo	movnf	01 0011	19	13 DC3	83 53	S		
		01 0100	20	14 DC4	84 54	T		
		01 0101	21	15 NAK	85 55	U		
		01 0110	22	16 SYN	86 56	V		
		01 0111	23	17 ETB	87 57	W		
mult	01 1000	24	18 CAN	88 58	X			
multu	01 1001	25	19 EM	89 59	Y			
div	01 1010	26	1a SUB	90 5a	Z			
divu	01 1011	27	1b ESC	91 5b	[
		01 1100	28	1c FS	92 5c	\		
		01 1101	29	1d GS	93 5d]		
		01 1110	30	1e RS	94 5e	^		
		01 1111	31	1f US	95 5f	_		
lb	add cvt.sf	10 0000	32	20 Space	96 60	-		
lh	addu cvt.dsf	10 0001	33	21 !	97 61	a		
lw	sub	10 0010	34	22 "	98 62	b		
lw	subu	10 0011	35	23 #	99 63	c		
lbu	and cvt.wf	10 0100	36	24 \$	100 64	d		
lhu	or	10 0101	37	25 %	101 65	e		
lwr	xor	10 0110	38	26 &	102 66	f		
	nor	10 0111	39	27 '	103 67	g		
sb		10 1000	40	28 (104 68	h		
sh		10 1001	41	29)	105 69	i		
swl	sit	10 1010	42	2a *	106 6a	j		
sw	situ	10 1011	43	2b +	107 6b	k		
		10 1100	44	2c ,	108 6c	l		
		10 1101	45	2d -	109 6d	m		
swr		10 1110	46	2e /	110 6e	n		
cache		10 1111	47	2f .	111 6f	o		
ll	tge c.fsf	11 0000	48	30 0	112 70	p		
lwc1	tgeu c.unsf	11 0001	49	31 1	113 71	q		
lwc2	tilt c.eqsf	11 0010	50	32 2	114 72	r		
pref	tltu c.ueqsf	11 0011	51	33 3	115 73	s		
ldc1	teq c.ofsf	11 0100	52	34 4	116 74	t		
ldc2	c.ulsf	11 0101	53	35 5	117 75	u		
tne	c.olef	11 0110	54	36 6	118 76	v		
	c.uolef	11 0111	55	37 7	119 77	w		
sc	c.ssf	11 1000	56	38 8	120 78	x		
swc1	c.nglef	11 1001	57	39 9	121 79	y		
swc2	c.seqf	11 1010	58	3a :	122 7a	z		
	c.nglf	11 1011	59	3b ;	123 7b	{		
	c.ltlf	11 1100	60	3c <	124 7c	-		
sdc1	c.ngef	11 1101	61	3d =	125 7d	}		
sdc2	c.lef	11 1110	62	3e >	126 7e	~		
	c.ngtf	11 1111	63	3f ?	127 7f	DEL		

(1) opcode(31:26) == 0

(2) opcode(31:26) == 17_{ten} (11_{hex}); if fmt(25:21)==16_{ten} (10_{hex}) f=s (single); if fmt(25:21)==17_{ten} (11_{hex}) f=d (double)

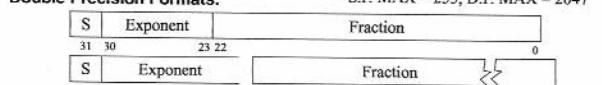
(3)

IEEE 754 FLOATING-POINT STANDARD

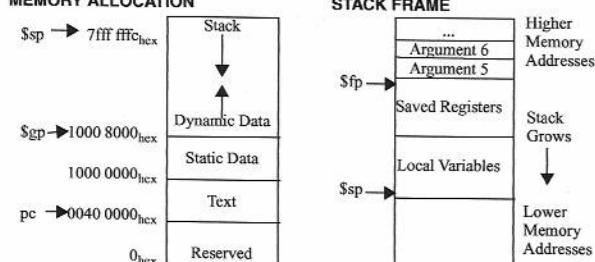
$$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

where Single Precision Bias = 127,
Double Precision Bias = 1023.

IEEE Single Precision and Double Precision Formats:



MEMORY ALLOCATION

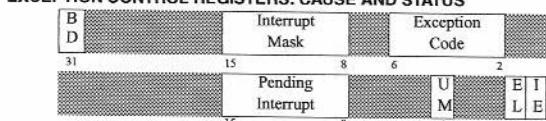


DATA ALIGNMENT

Double Word		Word	
Word		Word	
Halfword	Halfword	Halfword	Halfword
Byte	Byte	Byte	Byte
0	1	2	3
4	5	6	7

Value of three least significant bits of byte address (Big Endian)

EXCEPTION CONTROL REGISTERS: CAUSE AND STATUS



BD = Branch Delay, UM = User Mode, EL = Exception Level, IE = Interrupt Enable

EXCEPTION CODES

Number	Name	Cause of Exception	Number	Name	Cause of Exception
0	Int	Interrupt (hardware)	9	Bp	Breakpoint Exception
4	AdEL	Address Error Exception (load or instruction fetch)	10	RI	Reserved Instruction Exception
5	AdES	Address Error Exception (store)	11	CpU	Coprocessor Unimplemented
6	IBE	Bus Error on Instruction Fetch	12	Ov	Arithmetic Overflow Exception
7	DBE	Bus Error on Load or Store	13	Tr	Trap
8	Sys	System Call Exception	15	FPE	Floating Point Exception

SIZE PREFIXES (10^x for Disk, Communication; 2^x for Memory)

PRE-SIZE	FIX	PRE-SIZE	FIX	PRE-SIZE	FIX
10 ³ , 2 ¹⁰	Kilo-	10 ¹⁵ , 2 ³⁰	Peta-	10 ⁻³	milli-
10 ⁶ , 2 ²⁰	Mega-	10 ¹⁸ , 2 ⁶⁰	Exa-	10 ⁻⁶	micro-
10 ⁹ , 2 ³⁰	Giga-	10 ²¹ , 2 ⁷⁰	Zetta-	10 ⁻⁹	nano-
10 ¹² , 2 ⁴⁰	Tera-	10 ²⁴ , 2 ⁸⁰	Yotta-	10 ⁻¹²	zepto-

The symbol for each prefix is just its first letter, except μ is used for micro.

MIPS Reference Data Card ("Green Card") 1. Pull along perforation to separate card 2. Fold bottom side (columns 3 and 4) together