



Contact:  
Magnus Jahre (952 22 309)

## TDT4255 COMPUTER DESIGN EXAM

Saturday 3. December

Time: 09:00 – 12:00

ENGLISH

Allowed Aids:

D.

No written or handwritten examination support materials are permitted.

A specified, simple calculator is permitted.

*Use the provided space to answer the problems. If you need more space, an extra answer box is available on the last page of the test. The test accounts for 50% of the final grade, and the provided points show the maximal number of points that can be achieved on each assignment. Read the problem texts thoroughly. You can answer the questions in English or Norwegian.*

**Candidate Number:**

**Problem 1    Instruction sets (10 points)**

- a) (5 p) Translate the following C-code into MIPS instructions, making any necessary assumptions. The MIPS instruction reference can be found as an appendix.

```
if ( a > b ) a += 1;  
else a = 15;
```

Answer:

- b) (5 p) Explain how nested procedure calls can be supported, and give two examples of how the computer designer can make procedure calls more efficient.

Answer:

**Candidate Number:**

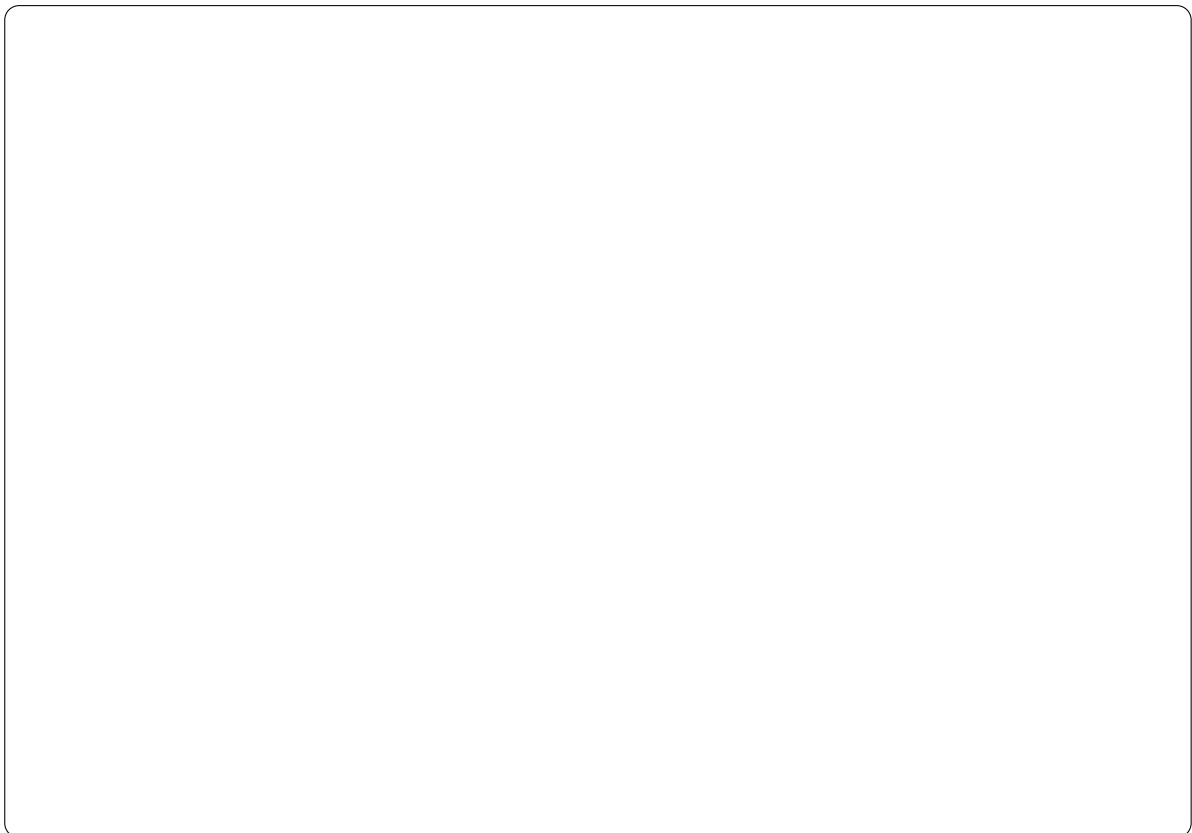
```
1 # assume that $s0 contains a vaild memory address
2 lw $t1, 0($s0)
3 ori $t2, $zero, 42
4 bne $t1, $t2, notequal
5     ori $s1, $zero, 1
6     jmp end
7 notequal:
8     ori $s1, $zero, 0
9 end:
```

Figure 1: MIPS Assembly Segment

**Problem 2 Single Cycle Processor (10 points)**

- a) (5 p) Draw a block diagram of a single cycle processor that can execute the MIPS program in Figure 1.

Answer:



**Candidate Number:**

b) (5 p) Translate the instructions on line 2 and line 3 into control words for your processor.

Answer:

**Problem 3    Pipelining (10 points)**

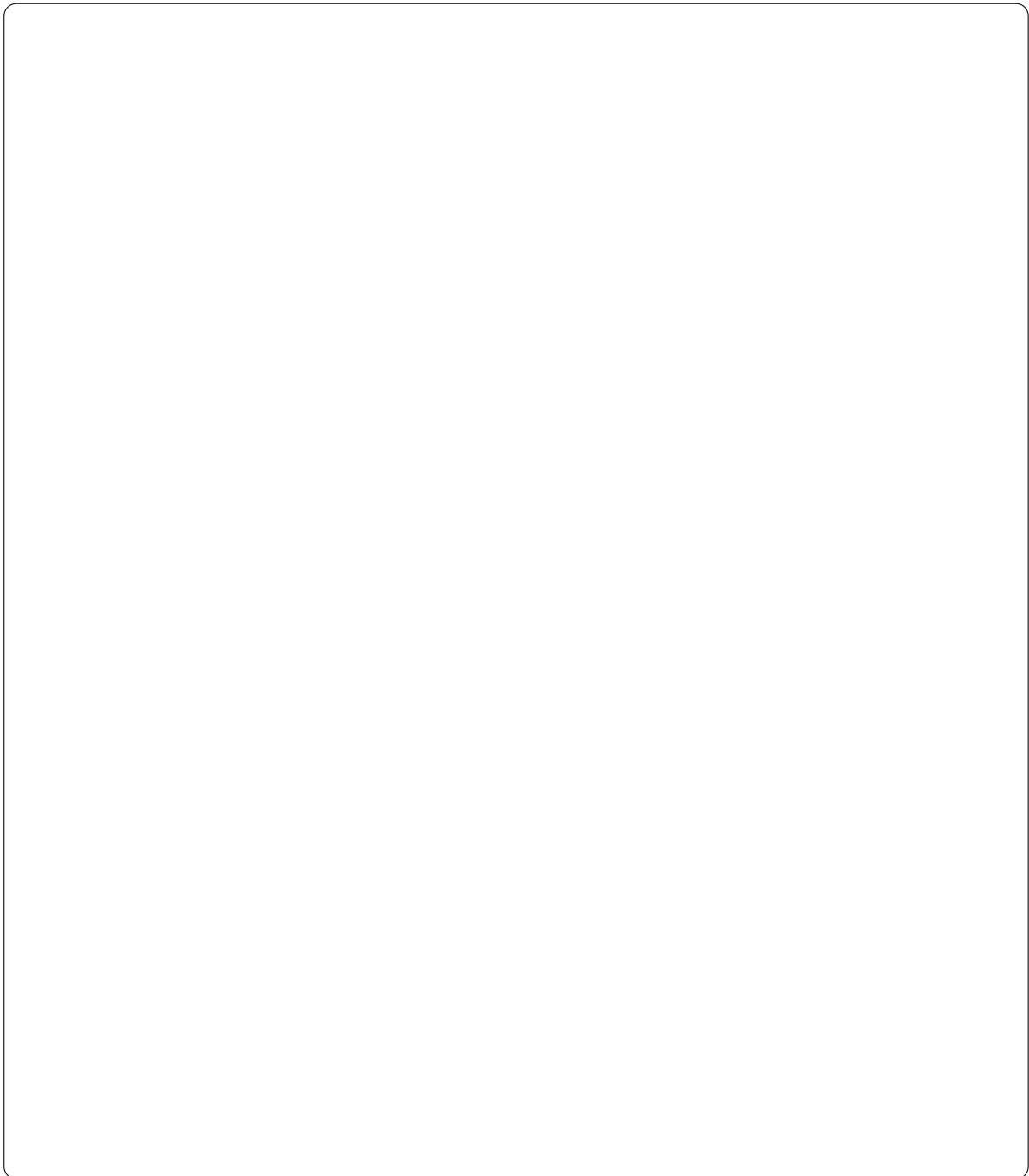
a) (5 p) Explain how *forwarding* can be used to avoid stalling on data hazards.

Answer:

**Candidate Number:**

- b) (5 p) Illustrate the execution of the code in Figure 1 on page 3 on a 5-stage pipeline given that the contents in the memory address in \$s0 is 42. Make any assumptions about the pipelined architecture that are necessary.

Answer:



**Candidate Number:**

**Problem 4    Instruction Level Parallelism (10 points)**

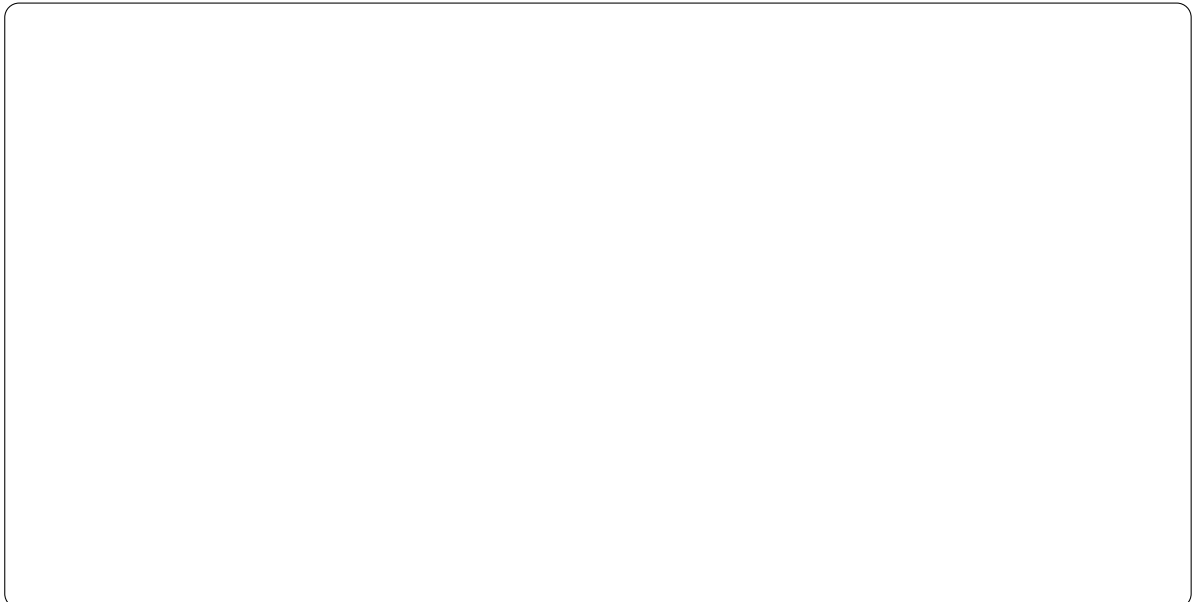
- a) (5 p) Explain the difference between a *dependency* and a *hazard*.

Answer:



- b) (5 p) Explain how a reorder buffer can be used to support speculative execution.

Answer:



**Candidate Number:**

**Problem 5 Memory Systems (10 points)**

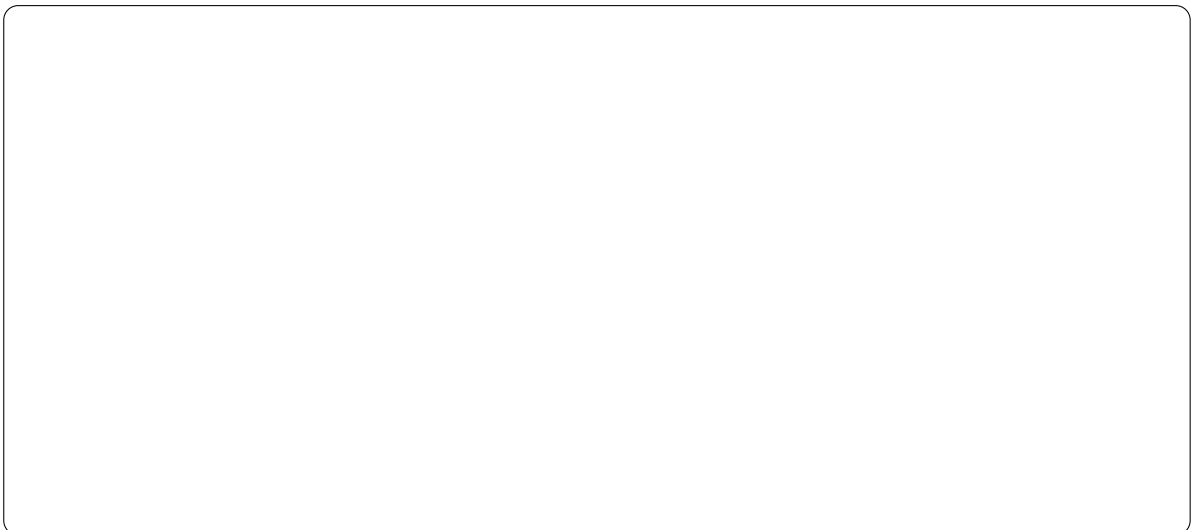
- a) (5 p) Draw a block diagram of a 2-way set associative cache.

Answer:



- b) (5 p) You have a 32b physical address, a 64B cache line, and an 8-way set associative cache. The cache size is 2MB. How many bits are needed for the block offset, the index, and the tag?

Answer:



**Candidate Number:**

**Additional Answer Space**

Answer:

A large, empty rectangular box with a thin black border, intended for the student to write their answer. The box is vertically oriented and occupies most of the page's width and height.

**Candidate Number:**



# MIPS Reference

## MIPS Reference Data

### CORE INSTRUCTION SET

NAME, MNEMONIC	FOR-MAT	OPERATION (in Verilog)	OPCODE / FUNCT (Hex)
Add	add R	$R[rd] = R[rs] + R[rt]$	(1) 0/20 <sub>hex</sub>
Add Immediate	addi I	$R[rt] = R[rs] + \text{SignExtImm}$	(1,2) 9 <sub>hex</sub>
Add Imm. Unsigned	addiu I	$R[rt] = R[rs] + \text{SignExtImm}$	(2) 9 <sub>hex</sub>
Add Unsigned	addu R	$R[rd] = R[rs] + R[rt]$	0/21 <sub>hex</sub>
And	and R	$R[rd] = R[rs] \& R[rt]$	0/24 <sub>hex</sub>
And Immediate	andi I	$R[rt] = R[rs] \& \text{ZeroExtImm}$	(3) 0 <sub>hex</sub>
Branch On Equal	beq I	$\text{if}(R[rs] == R[rt])$ $PC = PC + 4 + \text{BranchAddr}$	(4) 4 <sub>hex</sub>
Branch On Not Equal	bne I	$\text{if}(R[rs] != R[rt])$ $PC = PC + 4 + \text{BranchAddr}$	(4) 5 <sub>hex</sub>
Jump	j J	$PC = \text{JumpAddr}$	(5) 2 <sub>hex</sub>
Jump And Link	jal J	$R[31] = PC + 8; PC = \text{JumpAddr}$	(5) 3 <sub>hex</sub>
Jump Register	jr R	$PC = R[rs]$	0/08 <sub>hex</sub>
Load Byte Unsigned	lbu I	$R[rt] = \{24'b0, M[R[rs]] + \text{SignExtImm}(7:0)\}$	(2) 24 <sub>hex</sub>
Load Halfword Unsigned	lhu I	$R[rt] = \{16'b0, M[R[rs]] + \text{SignExtImm}(15:0)\}$	(2) 25 <sub>hex</sub>
Load Linked	ll I	$R[rt] = M[R[rs] + \text{SignExtImm}]$	(2,7) 30 <sub>hex</sub>
Load Upper Imm.	lui I	$R[rt] = \{\text{imm}, 16'b0\}$	6 <sub>hex</sub>
Load Word	lw I	$R[rt] = M[R[rs] + \text{SignExtImm}]$	(2) 23 <sub>hex</sub>
Nor	nor R	$R[rd] = \sim(R[rs]   R[rt])$	0/27 <sub>hex</sub>
Or	or R	$R[rd] = R[rs]   R[rt]$	0/25 <sub>hex</sub>
Or Immediate	ori I	$R[rt] = R[rs]   \text{ZeroExtImm}$	(3) 0 <sub>hex</sub>
Set Less Than	slt R	$R[rd] = (R[rs] < R[rt]) ? 1 : 0$	0/28 <sub>hex</sub>
Set Less Than Imm.	slti I	$R[rt] = (R[rs] < \text{SignExtImm}) ? 1 : 0$	(2) 9 <sub>hex</sub>
Set Less Than Imm. Unsigned	sltiu I	$R[rt] = (R[rs] < \text{SignExtImm}) ? 1 : 0$	(2,6) b <sub>hex</sub>
Set Less Than Unsig.	sltu R	$R[rd] = (R[rs] < R[rt]) ? 1 : 0$	(6) 0/29 <sub>hex</sub>
Shift Left Logical	sll R	$R[rd] = R[rt] \ll \text{shamt}$	0/00 <sub>hex</sub>
Shift Right Logical	srl R	$R[rd] = R[rt] \gg \text{shamt}$	0/02 <sub>hex</sub>
Store Byte	sb I	$M[R[rs] + \text{SignExtImm}(7:0)] = R[rt](7:0)$	(2) 28 <sub>hex</sub>
Store Conditional	sc I	$M[R[rs] + \text{SignExtImm}] = R[rt];$ $R[rt] = \{\text{atomic}\} ? 1 : 0$	(2,7) 38 <sub>hex</sub>
Store Halfword	sh I	$M[R[rs] + \text{SignExtImm}(15:0)] = R[rt](15:0)$	(2) 29 <sub>hex</sub>
Store Word	sw I	$M[R[rs] + \text{SignExtImm}] = R[rt]$	(2) 2b <sub>hex</sub>
Subtract	sub R	$R[rd] = R[rs] - R[rt]$	(1) 0/22 <sub>hex</sub>
Subtract Unsigned	subu R	$R[rd] = R[rs] - R[rt]$	0/23 <sub>hex</sub>

(1) May cause overflow exception  
 (2) SignExtImm = { 16{immediate[15]}, immediate }  
 (3) ZeroExtImm = { 16{1b'0'}, 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

### ARITHMETIC CORE INSTRUCTION SET

NAME, MNEMONIC	FOR-MAT	OPERATION	OPCODE / FUNCT (Hex)
Branch On FP True	bclt FI	$\text{if}(FPcond) PC = PC + 4 + \text{BranchAddr}$	(4) 11/8/1/-
Branch On FP False	bclt FI	$\text{if}(\sim FPcond) PC = PC + 4 + \text{BranchAddr}$	(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 Double	add.d FR	$\{F[fd], F[fd+1]\} = \{F[fs], F[fs+1]\} + \{F[ft], F[ft+1]\}$	11/11/-/0
FP Compare Single	c.x.s* FR	$FPcond = (F[fs] \text{ op } F[ft]) ? 1 : 0$	11/10/-/y
FP Compare Double	c.x.d* FR	$FPcond = (\{F[fs], F[fs+1]\} \text{ op } \{F[ft], F[ft+1]\}) ? 1 : 0$	11/11/-/y
FP Divide Single	div.s FR	$F[fd] = F[fs] / F[ft]$	11/10/-/3
FP Divide Double	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 Double	mul.d FR	$\{F[fd], F[fd+1]\} = \{F[fs], F[fs+1]\} * \{F[ft], F[ft+1]\}$	11/11/-/2
FP Subtract Single	sub.s FR	$F[fd] = F[fs] - F[ft]$	11/10/-/1
FP Subtract Double	sub.d FR	$\{F[fd], F[fd+1]\} = \{F[fs], F[fs+1]\} - \{F[ft], F[ft+1]\}$	11/11/-/1
Load FP Single	lwc1 I	$F[rt] = M[R[rs] + \text{SignExtImm}]$	(2) 31/-/-/4
Load FP Double	ldc1 I	$F[rt] = M[R[rs] + \text{SignExtImm}];$ $F[rt+1] = M[R[rs] + \text{SignExtImm} + 4]$	(2) 35/-/-/4
Move From Hi	mthi R	$R[rd] = Hi$	0/-/-/10
Move From Lo	mflr R	$R[rd] = Lo$	0/-/-/12
Move From Control	mfc0 R	$R[rd] = CR[rs]$	10/0/-/0
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	$R[rd] = R[rt] \gg \text{shamt}$	0/-/-/3
Store FP Single	swc1 I	$M[R[rs] + \text{SignExtImm}] = F[rt]$	(2) 39/-/-/3
Store FP Double	sdc1 I	$M[R[rs] + \text{SignExtImm}] = F[rt];$ $M[R[rs] + \text{SignExtImm} + 4] = F[rt+1]$	(2) 3d/-/-/3

\* (x is eq, lt, or le) (op is ==, <, or <=) (y is 32, 3c, or 3e)

### FLOATING-POINT INSTRUCTION FORMATS

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

### PSEUDOINSTRUCTION SET

NAME	MNEMONIC	OPERATION
Branch Less Than	blt	$\text{if}(R[rs] < R[rt]) PC = \text{Label}$
Branch Greater Than	bgt	$\text{if}(R[rs] > R[rt]) PC = \text{Label}$
Branch Less Than or Equal	bte	$\text{if}(R[rs] <= R[rt]) PC = \text{Label}$
Branch Greater Than or Equal	bge	$\text{if}(R[rs] >= R[rt]) PC = \text{Label}$
Load Immediate	li	$R[rd] = \text{immediate}$
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.
Sat	1	Assembler Temporary	No
Sv0-Sv1	2-3	Values for Function Results and Expression Evaluation	No
Sa0-Sa3	4-7	Arguments	No
St0-St7	8-15	Temporaries	No
Ss0-Ss7	16-23	Saved Temporaries	Yes
St8-St19	24-25	Temporaries	No
Sk0-Sk1	26-27	Reserved for OS Kernel	No
Sgp	28	Global Pointer	Yes
Ssp	29	Stack Pointer	Yes
Sfp	30	Frame Pointer	Yes
Sra	31	Return Address	Yes

Copyright 2009 by Elsevier, Inc., All rights reserved. From Patterson and Hennessy, *Computer Organization and Design*, 4th ed.

Candidate Number:

**OPCODES, BASE CONVERSION, ASCII SYMBOLS**

MIPS opcode (31:26)	(1) MIPS funct (5:0)	(2) MIPS funct (5:0)	Binary	Decimal	Hexa-decimal	ASCII Character	Decimal	Hexa-decimal	ASCII Character
(1)	sll	add.f	00 0000	0	0	NUL	64	40	@
		sub.f	00 0001	1	1	SOH	65	41	A
	srl	mul.f	00 0010	2	2	STX	66	42	B
	jal	div.f	00 0011	3	3	ETX	67	43	C
	beq	sllv	00 0100	4	4	EOT	68	44	D
	bne	abs.f	00 0101	5	5	ENQ	69	45	E
	blez	srlv	00 0110	6	6	ACK	70	46	F
	bgtz	sra	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
	slli	movz	00 1010	10	a	LF	74	4a	J
	slliu	movn	00 1011	11	b	VT	75	4b	K
	andi	syscall	00 1100	12	c	FF	76	4c	L
	ori	break	00 1101	13	d	CR	77	4d	M
	xori	trunc.w.f	00 1110	14	e	SO	78	4e	N
	lui	sync	00 1111	15	f	SI	79	4f	O
		floor.w.f	01 0000	16	10	DLE	80	50	P
(2)	mfi		01 0001	17	11	DC1	81	51	Q
	mfi		01 0010	18	12	DC2	82	52	R
	mfi		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	10 0000	32	20	Space	96	60	`
	lh	addu	10 0001	33	21	!	97	61	a
	lwl	sub	10 0010	34	22	"	98	62	b
	lwr	subu	10 0011	35	23	#	99	63	c
	lbu	and	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	sll	10 1010	42	2a	*	106	6a	j
	sw	sllv	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	rge	11 0000	48	30	0	112	70	p
	lwc1	rgeu	11 0001	49	31	1	113	71	q
	lwc2	tlr	11 0010	50	32	2	114	72	r
	pref	tlru	11 0011	51	33	3	115	73	s
	lwc1	teq	11 0100	52	34	4	116	74	t
	lwc2	tne	11 0101	53	35	5	117	75	u
			11 0110	54	36	6	118	76	v
			11 0111	55	37	7	119	77	w
	sc	c.srf	11 1000	56	38	8	120	78	x
	swc1	c.npl.f	11 1001	57	39	9	121	79	y
	swc2	c.seq.f	11 1010	58	3a	:	122	7a	z
		c.ngl.f	11 1011	59	3b	:	123	7b	{
		c.ltr.f	11 1100	60	3c	<	124	7c	
	sdc1	c.npl.f	11 1101	61	3d	=	125	7d	}
	sdc2	c.le.f	11 1110	62	3e	>	126	7e	~
		c.ngt.f	11 1111	63	3f	?	127	7f	DEL

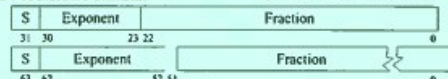
(1) opcode(31:26) == 0  
 (2) opcode(31:26) == 17<sub>hex</sub> (11<sub>hex</sub>); if fmt(25:21) == 16<sub>hex</sub> (10<sub>hex</sub>) f = s (single);  
 if fmt(25:21) == 17<sub>hex</sub> (11<sub>hex</sub>) f = d (double)

Copyright 2009 by Elsevier, Inc., All rights reserved. From Patterson and Hennessy, *Computer Organization and Design*, 4th ed.

**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:

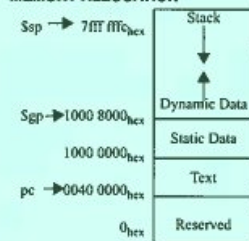


**IEEE 754 Symbols**

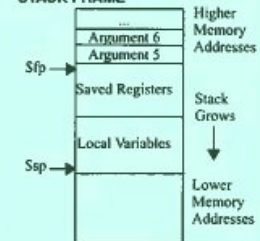
Exponent	Fraction	Object
0	0	$\pm 0$
0	$\neq 0$	$\pm \text{Denorm}$
1 to MAX - 1	anything	$\pm \text{Fl. Pt. Num.}$
MAX	0	$\pm \infty$
MAX	$\neq 0$	NaN

S.P. MAX = 255, D.P. MAX = 2047

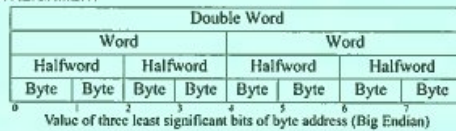
**MEMORY ALLOCATION**



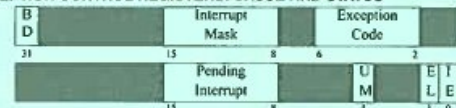
**STACK FRAME**



**DATA ALIGNMENT**



**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	R1	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	Syscall Exception	15	FPE	Floating Point Exception

**SIZE PREFIXES (10<sup>3</sup> for Disk, Communication; 2<sup>3</sup> for Memory)**

SIZE	PRE-FIX	SIZE	PRE-FIX	SIZE	PRE-FIX	SIZE	PRE-FIX
10 <sup>3</sup> , 2 <sup>10</sup>	Kilo-	10 <sup>15</sup> , 2 <sup>50</sup>	Peta-	10 <sup>3</sup>	milli-	10 <sup>-15</sup>	femto-
10 <sup>6</sup> , 2 <sup>20</sup>	Mega-	10 <sup>18</sup> , 2 <sup>60</sup>	Exa-	10 <sup>-6</sup>	micro-	10 <sup>-18</sup>	atto-
10 <sup>9</sup> , 2 <sup>30</sup>	Giga-	10 <sup>21</sup> , 2 <sup>70</sup>	Zetta-	10 <sup>-9</sup>	nano-	10 <sup>-21</sup>	zepto-
10 <sup>12</sup> , 2 <sup>40</sup>	Tera-	10 <sup>24</sup> , 2 <sup>80</sup>	Yotta-	10 <sup>-12</sup>	pico-	10 <sup>-24</sup>	yocto-

The symbol for each prefix is just its first letter, except  $\mu$  is used for micro.

Candidate Number: