OPCODES, BASE CONVERSION, ASCII SYMBOLS

③

MIPS opcode (31:26)	(1) MIPS funct (5:0)	(2) MIPS funct (5:0)	Binary	Decimal	Hexadecimal	ASCII Character	Decimal	Hexadecimal	ASCII Character
(1)	sll	add.*f*	00 0000	0	0	NUL	64	40	@
		sub.*f*	00 0001	1	1	SOH	65	41	A
j	srl	mul.*f*	00 0010	2	2	STX	66	42	B
jal	sra	div.*f*	00 0011	3	3	ETX	67	43	C
beq	sllv	sqrt.*f*	00 0100	4	4	EOT	68	44	D
bne		abs.*f*	00 0101	5	5	ENQ	69	45	E
blez	srlv	mov.*f*	00 0110	6	6	ACK	70	46	F
bgtz	srav	neg.*f*	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	round.w.*f*	00 1100	12	c	FF	76	4c	L
ori	break	trunc.w.*f*	00 1101	13	d	CR	77	4d	M
xori		ceil.w.*f*	00 1110	14	e	SO	78	4e	N
lui	sync	floor.w.*f*	00 1111	15	f	SI	79	4f	O
	mfhi		01 0000	16	10	DLE	80	50	P
(2)	mthi		01 0001	17	11	DC1	81	51	Q
	mflo	movz.*f*	01 0010	18	12	DC2	82	52	R
	mtlo	movn.*f*	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.s.*f*	10 0000	32	20	Space	96	60	`
lh	addu	cvt.d.*f*	10 0001	33	21	!	97	61	a
lwl	sub		10 0010	34	22	"	98	62	b
lw	subu		10 0011	35	23	#	99	63	c
lbu	and	cvt.w.*f*	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	slt		10 1010	42	2a	*	106	6a	j
sw	sltu		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.*f*.*f*	11 0000	48	30	0	112	70	p
lwc1	tgeu	c.un.*f*	11 0001	49	31	1	113	71	q
lwc2	tlt	c.eq.*f*	11 0010	50	32	2	114	72	r
pref	tltu	c.ueq.*f*	11 0011	51	33	3	115	73	s
	teq	c.olt.*f*	11 0100	52	34	4	116	74	t
ldc1		c.ult.*f*	11 0101	53	35	5	117	75	u
ldc2	tne	c.ole.*f*	11 0110	54	36	6	118	76	v
		c.ule.*f*	11 0111	55	37	7	119	77	w
sc		c.sf.*f*	11 1000	56	38	8	120	78	x
swc1		c.ngle.*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.lt.*f*	11 1100	60	3c	<	124	7c	|
sdc1		c.nge.*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_{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)

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.

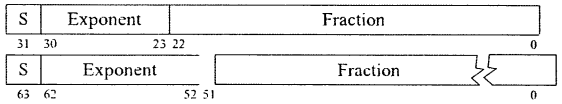
④

IEEE 754 Symbols

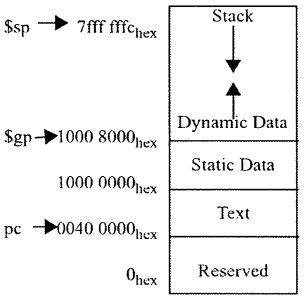
Exponent	Fraction	Object
0	0	± 0
0	≠0	± Denorm
1 to MAX - 1	anything	± Fl. Pt. Num.
MAX	0	±∞
MAX	≠0	NaN

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

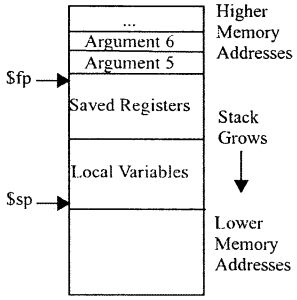
IEEE Single Precision and Double Precision Formats:



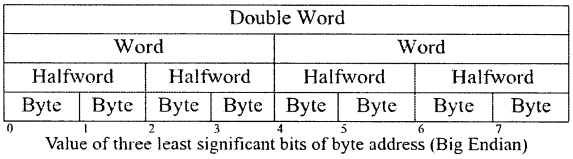
MEMORY ALLOCATION



STACK FRAME

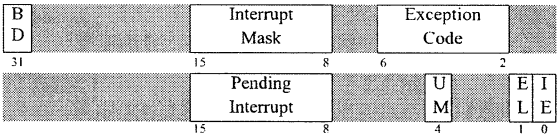


DATA ALIGNMENT



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	Syscall Exception	15	FPE	Floating Point Exception

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

SIZE	PREFIX	SIZE	PREFIX	SIZE	PREFIX	SIZE	PREFIX
10^3, 2^{10}	Kilo-	10^{15}, 2^{50}	Peta-	10^{-3}	milli-	10^{-15}	femto-
10^6, 2^{20}	Mega-	10^{18}, 2^{60}	Exa-	10^{-6}	micro-	10^{-18}	atto-
10^9, 2^{30}	Giga-	10^{21}, 2^{70}	Zetta-	10^{-9}	nano-	10^{-21}	zepto-
10^{12}, 2^{40}	Tera-	10^{24}, 2^{80}	Yotta-	10^{-12}	pico-	10^{-24}	yocto-

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

performance = clockrate / CPI

No Cycles = time x clockrate

$$\text{time} = \frac{\text{no instructions} \times \text{CPI}}{\text{clock rate}}$$

$$\text{no instructions} = \frac{\text{no cycles}}{\text{CPI}}$$

$$\text{clock rate} = \frac{\text{no instr.} \times \text{CPI}}{\text{time}}$$

$$\text{instructions per cycle} = \frac{1}{\text{CPI}} = \frac{\text{no instr.}}{\text{time} \times \text{clockrate}}$$

$$\frac{\text{Time new}}{\text{Time old}} \text{ so } \frac{f_{new}}{f_{old}}$$

$$\text{CPI} = \frac{\text{time} \times \text{clockrate}}{\text{no instructions}}$$

$$\text{Execution Time} = \frac{\text{no instr} \times \text{cycles}}{\text{clock rate}}$$

$$\text{capacitic load} = \frac{\text{Power}}{\text{volts}^2} \times \text{clock rate}$$

- increasing clock cycle time lowers clock rate,

$$\text{- Speedup} = \frac{\text{time old}}{\text{time new}} = \%$$

Amdahls Law

$$\text{Speedup} = \frac{\text{old}}{\text{unmodified} + \text{modified}}$$

$$\text{cycle Time} = \frac{1}{\text{rate}} \qquad 1\text{hz} = 1 \text{ cycle per second.}$$

16 bit single point floating Point

19.03125 = 10011.00001

$1.00110\,00001 \times 10^{4}$ $\quad y = \epsilon - 15$, $\epsilon = 19$

0 10011 0011000010

Guard → Round → sticky (shifted off bits.)

multiplication table

5x6 101 x 110

		Pro 0	110	110	11110	11110 = 30
mul 101	mcd 110	1100	11000	110000	= 9x5	
	010		001	000		

Find clock rate for P2 that reduces its execution time to a P1?, also no instruction

P1 Time = 7s $\quad \frac{2.5}{7} = 3.5\text{ghz}$ to reduce to a P1

P2 Time = 10s

P2 clock = 2.5ghz

P2 instructions = 30×10^9 $\quad \frac{30\times10^9}{\frac{7}{10}}$

A	$10^6 \times .1$	pico	10^{-12}
B	$10^6 \times .2$	nano	10^{-9}
C	$10^6 \times .5$	micro	10^{-6}
D	$10^6 \times .2$	milli	10^{-3}

Given a program with 10^6 instructions divided as follows:

	A	B	C	D
2.5ghz	10%	0%	50%	20%
CPI	1	2	3	3

Time A: $1\times10^5 \times 1 / 2.5\times10^9 = .4 \times 10^{-4}$

Time B: $2\times10^5 \times 2 / 2.5\times10^9 = 1.6 \times 10^{-4}$

Time C: $5\times10^5 \times 3 / 2.5\times10^9 = 6.0 \times 10^{-4}$

Time D: $2\times10^5 \times 3 / 2.5\times10^9 = 2.4 \times 10^{-4}$

Global CPI $\quad \frac{\text{time} \times \text{clock rate}}{\text{no inst}}$ $\quad \frac{10.4\times10^{-4} \times 2.5\times10^9}{10^6}$

What is execution time of program on 2 ghz machine?

Arith 1 cycle	650 instr	650×1
Store 5 cycles	100 "	100×5
load 5 cycles	600 "	600×5
Branch 2 cycles	50 "	50×2 +
	2.0×10^9	$= 2,125$ ns

Find clock cycles required?, add up CPI x no instructions for the program.

Find CPI for same program?,

$$\frac{(2.125\times10^{-6}) \times (2.0\times10^9)}{\text{total instructions (1000)}}$$

Geometric Mean / clock rate ratio: first, find all the ratios. Take next gen / old gen. Add up all the ratios, and raise that to inverse of count of ratios calculated. If 7 ratios, $\frac{1}{7}$ (sum of ratios).

Largest Relative Change

$$\frac{\text{largest ratio}}{\text{calculated.}}$$

Largest relative Change in voltage

older

younger

Find average CPI

Arith	70%	2 cycles
Load/store	10%	6 cycles
Branch	20%	3 cycles

$2 \times .7 + 6 \times .1 + 3 \times .2 = 2.6$

Floating Point Addition

make exponents same, add values up, use 2's comp if necessary. Drop last carry bit, if any.

19.03125 to single precision floating pt,

19 / 2	R = 1	.03125 x 2	0
9 / 2	R = 1	.0625 x 2	0
4 / 2	R = 0	.125 x 2	0
2 / 2	R = 0	.25 x 2	0
1 / 2	R = 1	.5 x 2	1
0 / 2 done			

00001

10011

$10011.00001 = 1.00110\,00001 \times 2^4$

$y = \epsilon - 127$

$\epsilon = 131$

0 10000011 00110000100000000000000

Floating point multiplication

- add exponents.
- multiply values.
- count number of digits after . for both values. Shift final . by that amount of digits
- check sign based on originals.

non pipelined arch running at 2.5 Ghz that takes 5 cycles to finish an instruction, you want 5 pipelined stages. Now running at 2Ghz

- memory 30% of instructions, stall of 50 cycles happens 2%
- branch 20% of instructions, stall of 2 cycles, happens 70%

What is speedup?

$$\text{Speedup} = \frac{5}{2.5} \Big/ \frac{1 + .3 \times .02 \times 50 + .2 \times .2 \times 2}{2}$$

$$\text{Speedup} = \frac{5}{1 + .3 + .08} \times \frac{2}{2.5}$$

$$= \frac{4}{1.38}$$

3.75 x 29.625 in Floating Point

$3.75 = 11.11 = 1.111 \times 2^1$

$29.625 = 11101.101 = 1.1101101 \times 2^4$

multiply

```
1.1101101
0.1110111
0.0111011
0.0011101
-----------
11.0111100011
```

1.10111100011×2^1

$4+1+1 = 6$ $\quad 6 = E - 127$ $\quad E = 133$

0 10000101 1011 1100 0110 0000 0000 000

7-8 conditional sum

7 = 0111 $\quad -8 = 1000$

0	1	1	1
+1	0	0	0
(10,01)	(10,01)	(10,01)	(10,01)
(100,011)		(100,011)	
(10000, 01111)			

since addition, no carry in so solution is

0 | 1111

or −1

−3x − 6 by booths

$x = -3 = 1101$ $\quad y = -6 = 1010$

$-y = 6 = 0110$

U	V	X	X-1	
0000	0000	1101	0	
0110	0000	1101	0	
0011	0000	1110	1	
1101	0000	1110	1	
1110	1000	0111	0	
0100	1000	0111	0	
0010	0100	1011	1	
0001	0010	1101	1	

00010010 = 18

1.5 Ghz no pipeline, instruction buffers that negate instruction load penalties 80% of the time.

32kb L1 cache, 1ns access, 4 way associative, write through, 2% miss rate for data and 1/60 for instructions.

1mb L2 cache, 12ns access, 4 way ~~associative~~, write back, write-allocate, 30% miss rate, 20% dirty.

16b ram, 50ns access

RAW - read after write

when a read instruction occurs before a previous write instruction has finished.

WAR - write after read

When a write instruction occurs before a previous Read instruction has finished.

WAW - write after write

when a write instruction occurs before a previous write instruction has finished.

what is the effective access time for ldinq instructions?

$E\text{Atloads} = MR_{buffer}(T_{L1} + MR_{L1}(T_{L2} + MR_{L2}(1+dirty)T_{ram}))$

$= .2(1 + 1/60(12 + .3(1+.2)50))$

$= .3ns$

Branch delay slot?

No, a branch delay slot is only useful to reduce the penalty of address jumps in a pipelined machine. In a pipelined machine, when the branch occurs, two or three instructions following the branch are typically already loaded. These instruction must be flushed, reducing the efficiency of the pipeline. The branch delay reduces the penalty by one.

```
ld.b R2, 0(R3)   1
add R3, R3, 4    2     War with 1
mul R2, R2, R2   3     Raw with 1
st.b R2, 0(R3)   4     Raw with 2,3
sub R2, R3, R4   5     Raw with 2, war 4
bnez R2, square  6     Raw with 5
```

CPI Penalty

$\text{Penalty}_{CPI} = E\text{at loads} \times CK$

$= .3ns \times 1.5\,Ghz$

$= .45$ cycles

Recommended changing to make faster.

Two thirds of the penalty is due to the access time of level one cache. I would look at trying to reduce or negate this first.

1. add $t0, $s6, $s0	temp0 = A+f	f, g, h, i, j $s0 to $s4
2. add $t1, $s7, $s1	temp1 = B+g	A = $s6
3. lw $t0, 0($t0)	f = A[f]	B = $s7
4. addi $t2, $t0, 4	temp2 = temp0 + 1	
5. lw $t0, 0($t2)	temp0 = &temp2	f = A[f]
6. add $t0, $t0, $s0	temp0 = temp0 + f	T0 = f + A[f+1]
7. sw $t2, 0($t1)	temp2 = *temp1	B[G5] = A[f+1]

$s6 = 16 f = 8

$t0 = 24

(a) Block offset size = 2^{15}, 32k 2^{34} ram tag 25 bits

(b) $2^{40} - 2^{15} = 2^{25}$ 2^{25} pages × 64 words = $2^{25} \times 2^{6} = 2^{31}$ bytes.

(c) yes, because pages aren't written back to disk right away.
yes, need valid to determine if entry in page is valid

(d) cache registers should contain actual addresses so no translation is necessary to convert from virtual to physical, cache should map to physical memory

(e) setup a TLB, transition lookaside buffer so that you will have part of the page table locally, and wouldn't need to go to memory to get page table into.