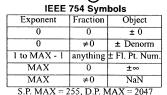
			SI	ON, A	SCII				3	
	(1) MIPS				Deci-		ASCII	Deci-	Hexa-	ASCI
pcode	funct	funct	Bi	nary	mal	deci-	Char-	mal	deci-	Char-
31:26)	(5:0)	(5:0)			mai	mal	acter	mai	mal	acter
(1)	sll	add.f	00	0000	0	0	NUL	64	40	(a)
		${ t sub} f$	00	1000	1	1	SOH	65	41	Ā
j .	srl	${\tt mul}f$	00	0010	2	2	STX	66	42	В
jal	sra	div.f	00	0011	3	3	ETX	67	43	C
bec	silv	sgrt.f	00	0100	4	4	EOT	68	44	D
bne		abs.f	00	0101	5	5	ENO	69	45	Ε
blez	srlv	mov.f		0110	6	6	ACŘ	70	46	F
bgtz	grav	neg f		0111	7	7	BEL	71	47	G
addi	fr			1000	8	8	BS	72	48	H
addiu	jalr			1001	9	9	HT	73	49	ï
slti	DOVZ			1010	10	a	LF	74	4a	Ĵ
sltiu	movn			1011	11	b	VT	75	4b	K
andi	syscall	round.w.f		1100	12	c	FF	76	4c	- î
ori	break	trunc.w.f		1101	13	d	CR	77	4d	M
xori	DICAN			1110	14	e	SO	78	4e	N
		ceil.w.f		1111	15	f	SI	79	4f	O
lui	sync	floor.w.f								
(2)	mfhi			0000	16	10	DLE	80	50	P
(2)	mthi	,		0001	17	11	DC1	81	51	Q
	mflo	movz.f		0010	18	12	DC2	82	52	R
	mtlo	movr.f		0011	19	13	DC3	83	53	S
				0100	20	14	DC4	84	54	T
				0101	21	15	NAK	85	55	U
				0110	22	16	SYN	86	56	V
				0111	23	17	ETB	87	57	W
	mult			1000	24	18	CAN	88	58	X
	multu			1001	25	19	EM	89	59	Y
	div		01	1010	26	la	SUB	90	5a	Z
	divu		01	1011	27	lb	ESC	91	5b	[
			01	1100	28	lc	FS	92	5c	7
			01	1101	29	1 d	GS	93	5d]
			01	1110	30	1e	RS	94	5e	ž
				1111	31	1 f	US	95	5f	
				0000	32	20	Space	96		
1b	acc								60	٠
lb lh	add addit	cvt.s. f					Space 1		60 61	
lh	addu	cvt.s.j	10	0001	33	21		97	61	a
lh lwl	addu sub		10 10	$0001 \\ 0010$	33 34	21 22	!	97 98	61 62	a b
lh lwl lw	addu sub subu	cvt.d f	10 10 10	0001 0010 0011	33 34 35	21 22 23	! " #	97 98 99	61 62 63	a b c
lh lwl lw lbu	addu sub subu and		10 10 10	0001 0010 0011 0100	33 34 35 36	21 22 23 24	! # S	97 98 99 100	61 62 63	a b c
lh lwl lw lbu lhu	addu sub subu and or	cvt.d f	10 10 10 10	0001 0010 0011 0100 0101	33 34 35 36 37	21 22 23 24 25	! # \$ %	97 98 99 100 101	61 62 63 64 65	a b c d
lh lwl lw lbu lhu	addu sub subu and or xor	cvt.d f	10 10 10 10 10	0001 0010 0011 0100 0101 0110	33 34 35 36 37 38	21 22 23 24 25 26	! # S	97 98 99 100 101 102	61 62 63 64 65 66	a b c d e f
lh lwl lw lbu lhu lwr	addu sub subu and or	cvt.d f	10 10 10 10 10 10	0001 0010 0011 0100 0101 0110	33 34 35 36 37 38 39	21 22 23 24 25 26 27	! # \$ % &	97 98 99 100 101 102 103	61 62 63 64 65 66 67	a b c d e f
lh lwl lw lbu lhu lwr	addu sub subu and or xor	cvt.d f	10 10 10 10 10 10 10	0001 0010 0011 0100 0101 0110 0111	33 34 35 36 37 38 39 40	21 22 23 24 25 26 27 28	! # \$ % &	97 98 99 100 101 102 103	61 62 63 64 65 66 67 68	a b c d e f g
lh lwl lbu lhu lwr	addu sub subu and or xor	cvt.d f	10 10 10 10 10 10 10	0001 0010 0011 0100 0101 0110 0111 1000 1001	33 34 35 36 37 38 39 40 41	21 22 23 24 25 26 27 28 29	# 	97 98 99 100 101 102 103 104 105	61 62 63 64 65 66 67 68 69	a b c d e f g h i
lh lwl lw lbu lhu lwr sb sh swl	addu sub subu and or xor nor	cvt.d f	10 10 10 10 10 10 10 10	0001 0010 0011 0100 0101 0110 0111 1000 1001	33 34 35 36 37 38 39 40 41 42	21 22 23 24 25 26 27 28 29 2a	! # \$ % &, ()	97 98 99 100 101 102 103 104 105 106	61 62 63 64 65 66 67 68 69 6a	a b c d e f g h i j
lh lwl lw lbu lhu lwr sb sh swl	addu sub subu and or xor	cvt.d f	10 10 10 10 10 10 10 10	0001 0010 0011 0100 0101 0110 0111 1000 1001 1010	33 34 35 36 37 38 39 40 41 42 43	21 22 23 24 25 26 27 28 29 2a 2b	# 	97 98 99 100 101 102 103 104 105 106 107	61 62 63 64 65 66 67 68 69 6a 6b	a b c d e f g h i
lh lwl lw lbu lhu lwr sb sh swl	addu sub subu and or xor nor	cvt.d f	10 10 10 10 10 10 10 10 10 10	0001 0010 0011 0100 0101 0111 1000 1001 1010 1011	33 34 35 36 37 38 39 40 41 42 43	21 22 23 24 25 26 27 28 29 2a 2b	# 5 % & ,	97 98 99 100 101 102 103 104 105 106 107	61 62 63 64 65 66 67 68 69 6a 6b	a b c d e f g h i j k
lh lwl lw lbu lhu lwr sb sh swl	addu sub subu and or xor nor	cvt.d f	10 10 10 10 10 10 10 10 10 10 10	0001 0010 0011 0100 0101 0110 0111 1000 1001 1011 1100 1101	33 34 35 36 37 38 39 40 41 42 43 44 45	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d	# 5 % &, () *	97 98 99 100 101 102 103 104 105 106 107 108 109	61 62 63 64 65 66 67 68 69 6a 6b 6c 6d	a b c d e f g h i j k l m
lh lwl lw lbu lhu lwr sb sh swl sw	addu sub subu and or xor nor	cvt.d f	10 10 10 10 10 10 10 10 10 10 10 10	0001 0010 0011 0100 0101 0111 1000 1001 1010 1110 1110	33 34 35 36 37 38 39 40 41 42 43 44 45 46	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2c 2d 2c 2d 2c 2d 2c 2d 2c 2d 2c 2d 2c 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d	! # 5 % & , () * +	97 98 99 100 101 102 103 104 105 106 107 108 109 110	61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e	a b c d e f g h i j k l m n
lh lwl lw lbu lhu lwr sb sh swl sw	addu sub subu and or xor nor	cvt.w/	10 10 10 10 10 10 10 10 10 10 10 10 10	0001 0010 0011 0100 0101 0111 1000 1001 1010 1110 1110	33 34 35 36 37 38 39 40 41 42 43 44 45 46 47	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f	! "# 5 % & () * +	97 98 99 100 101 102 103 104 105 106 107 108 109 110	61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f	a b c d e f g h i j k l m n o
lh lwl lw lbu lhu lwr sb sk sw sw sw cache	addu sub subu subu subu and or xor nor sit situ	cvt.wf	10 10 10 10 10 10 10 10 10 10 10 10 10 1	0001 0010 0011 0100 0101 0111 1000 1011 1110 1110 1111 0000	33 34 35 36 37 38 39 40 41 42 43 44 45 46 47	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30	! "# \$ % & & , () , + + . , 0	97 98 99 100 101 102 103 104 105 106 107 108 109 110 111	61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70	a b c d e f g h i j k l m n o
lh lwl lw lbu lhu sb sh swl sw swr cache ll lwc1	addu sub subu subu subu and or xor nor sit situ	cvt.wf	10 10 10 10 10 10 10 10 10 10 10 10 10 1	0001 0010 0011 0100 0101 0110 0111 1000 1011 1110 1110 1111 0000 0001	33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31	! ## \$ % & &	97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112	61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71	a b c d e f g h i j k l m n o p q
lh lwl lwl lbu lhu sb sh swl sw swr cache ll lwc1	addu sub subu subu subu and or xor nor sit situ	cvt.df cvt.wf c.ff c.unf c.eqf	10 10 10 10 10 10 10 10 10 10 10 10 11 11	0001 0010 0011 0100 0101 0110 0111 1000 1011 1110 1110 1111 0000 0001 0010	33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2c 30 31 32	! ## \$ % & &	97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112	61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 70 71 72	a b c d e f g h i j k l m n o
1h 1w1 1w 1bu 1hu 1hu sb sh swl sw swr cache 11 1wc1 1wc2	addu sub subu subu subu and or xor nor sit situ	cvt.wf	10 10 10 10 10 10 10 10 10 10 10 11 11 1	0001 0010 0011 0100 0101 0110 0111 1000 1011 1110 11110 11111 0000 0001 0011	33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33	! "# \$ % & , ()) * + - - - - 1 2 3	97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115	61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 71 72 73	a b c d e f g h i j k l m n o p q
lh lwl lwl lbu lhu sb sh swl sw swr cache ll lwc1	addu sub subu subu subu and or xor nor sit situ	cvt.df cvt.wf c.ff c.unf c.eef c.uef c.oltf	10 10 10 10 10 10 10 10 10 10 10 11 11 1	0001 0010 0011 0100 0101 0110 0111 1000 1011 1110 1110 1111 0000 0001 0010	33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33	! ## S % &	97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112	61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 70 71 72	a b c d e f g h i j k l m n o p q r
lh lwl lw lbu lhu lhu sb sk sw swr cache ll lwc2 pref	addu sub subu and or xor nor sit situ	cvt.wf cvt.wf c.eqf c.eqf c.eqf	10 10 10 10 10 10 10 10 10 10 10 10 11 11	0001 0010 0011 0100 0101 0110 0111 1000 1011 1110 11110 11111 0000 0001 0011	33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33	! "# # \$ % & &	97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115	61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 71 72 73	a b c d e f g h i j k l m n o p q r s
Ih Iwl Iwl Ibu Ihu Ihu Isb swl sw swr cache Il Iwc1 Iwc2 pref	addu sub subu and or xor nor sit situ	cvt.df cvt.wf c.ff c.unf c.eef c.uef c.oltf	10 10 10 10 10 10 10 10 10 10 10 10 11 11	0001 0010 0100 0101 0101 0111 1000 1011 1110 1110 0000 0001 0010 0010	33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33	! ## S % &	97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115	61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73	a b c c d e e f g h i j j k l m n o o p q r s t
Ih lwl lw lbu lhu lwr sb sw swr cache ll lwc1 lwc2 pref	addu sub subu subu subu subu subu subu su	cvt.df cvt.wf c.ef c.unf c.eef c.ueff c.ultf c.olef	10 10 10 10 10 10 10 10 10 10 10 11 11 1	0001 0010 0110 0100 0101 01101 1000 1011 1100 1011 1110 1110 11110 0000 0001 0010 0100 0101	33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34	! "# # \$ % & &	97 98 99 100 101 102 103 104 105 106 107 108 110 111 112 113 114 115 116 117	61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75	a b c c d e f ge h i j j k l m n o p q r s t u
Ih lw1 lw lbu lhu lhu sb sh sw swr cache ll lwc2 pref	addu sub subu subu subu subu subu subu su	cvt.df cvt.wf c.e.f.f c.ur.f c.eqf c.eqf c.oltf c.olef c.olef	10 10 10 10 10 10 10 10 10 10 10 11 11 1	0001 0010 0100 0101 0100 0101 1001 100	33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36	! "# \$ % & & , () * + + , , , , , , , , , , , , , , , , ,	97 98 99 100 101 102 103 104 105 106 107 108 110 111 112 113 114 115 116 117 118	61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 67 70 71 72 73 74 75	a b c c d e f g h i j j k l m n o p q r r s t u v w
Ih lw1 lw lbu lhu lhu sb sh swl sw swr cache ll lwc1 lwc2 pref	addu sub subu subu subu subu subu subu su	cvt.df cvt.wf c.ff c.unf c.eqf c.utef c.ultf c.ultf c.ulef c.ulef	10 10 10 10 10 10 10 10 10 10 10 10 11 11	0001 0010 0011 0100 0101 0110 0110 1001 1101 1101 1110 0000 0011 0100 0101 0101 0111 1100	33 34 35 36 37 38 39 40 41 42 43 44 45 56 51 52 53 54 55 56	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2c 2d 31 31 33 34 35 36 37 37 38	! "# \$ % & () * + - / 0 1 2 3 4 5 6 7 8	97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 119	61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6c 6f 70 71 72 73 74 75 76	a b c c d e e f ge h i j j k l m n o o p q q r s t u v v w x x
Ih lw1 lw lbu lhu lhu sb sh sw swr cache ll lwc1 lwc2 pref ldc1 ldc2 sc swc1	addu sub subu subu subu subu subu subu su	cvt.df cvt.wf c.sf c.unf c.ecf c.uedf c.ultf c.olef c.ulef c.sif c.nglef	10 10 10 10 10 10 10 10 10 10 10 10 10 1	0001 0010 0101 0100 0111 1000 1101 1010 1101 1111 1110 0000 0011 0100 0101 0110 0111 1100 0101 0101 0110	33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 30 31 32 33 34 35 36 37 38 39 39 39 39 30 30 30 30 30 30 30 30 30 30 30 30 30	! "# \$ % & & ()) * + + / 0 1 1 2 3 4 4 5 6 6 7 7 8 9	97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 111 112 113 114 115 116 117 118 119 119 110 1110 1111 1117 1118 1119 1119 1119 1119 1119	61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78 79	a b c c d e e f g h i j j k l m n o o p q r s s t u v w w x y
Ih lw1 lw lbu lhu lhu sb sh swl sw swr cache ll lwc1 lwc2 pref	addu sub subu subu subu subu subu subu su	cvt.df cvt.wf cvt.wf c.ef c.vef c.vef c.vetf c.olef c.ulef c.sef c.secf	10 10 10 10 10 10 10 10 10 10 10 10 10 1	0001 0010 0011 0100 0101 0110 1000 1011 1100 1101 1100 0011 0100 0010 0111 0100 1101 0110 0111 0110 0111 0110 0111	33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 47 50 51 52 53 53 54 55 55 56 57 58	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37 38 39 39 39 39 39 39 39 39 39 39 39 39 39	! "# \$ % & () * + - / 0 1 2 3 4 5 6 7 8	97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 121	61 62 63 64 65 66 67 68 69 60 60 60 60 60 67 71 72 73 74 75 76 77 78 79 72	a b c c d e e f g h i j j k l m n o o p q r r s t u v w w x y z
Ih lw1 lw lbu lhu lhu sb sh sw swr cache ll lwc1 lwc2 pref ldc1 ldc2 sc swc1	addu sub subu subu subu subu subu subu su	cvt.df cvt.wf c.ff c.urf c.eqf c.ueff c.ultf c.ultf c.sif c.nglef c.secf c.nglf	10 10 10 10 10 10 10 10 10 10 10 11 11 1	0001 0010 0011 0100 0101 0110 1001 1011 1100 1101 1110 0000 0011 0100 0110 0111 1100 1101 0110 0111 0110 0111	33 34 35 36 37 38 39 40 41 42 43 44 45 50 50 51 52 53 54 55 57 58 59	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 30 31 32 26 33 34 35 36 37 38 39 3a 3b	# \$ \$ % & & ; () ** + + , ** / / 0 1 2 2 3 4 4 5 6 6 7 7 8 9 9 : ; ;	97 98 99 100 101 102 103 104 105 106 107 108 110 111 112 113 114 115 116 117 118 119 120 121 121	61 62 63 63 64 65 66 66 67 70 71 72 73 74 75 76 77 78 79 72 73 74 75 75 75 75 75 75 75 75 75 75 75 75 75	a b c c d e e f g h i j j k l m n o o p q r s s t u v w w x y
Ih lw1 lw lbu lhu lhu sb sh swl swl cache ll lwc1 lwc2 pref ldc1 ldc2 sc swc1 swc2	addu sub subu subu subu subu subu subu su	cvt.df cvt.wf c.ff c.urf c.eqf c.uedf c.ultf c.ultf c.olef c.ulef c.sif c.nglef c.nglf c.nglf c.nglf	10 10 10 10 10 10 10 10 10 10 10 11 11 1	0001 0010 0011 0110 0101 0101 1001 100	33 34 35 36 37 38 39 40 41 42 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 60	21 22 23 24 25 26 27 27 28 29 2a 2c 2d 2e 2f 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3c	! "# \$ % & & , ()) * + + , , / 0 1 2 3 4 5 5 6 7 7 8 8 9 : ; ; <	97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 121 122 123 124	61 62 63 64 65 66 67 68 69 6a 66 66 67 70 71 72 73 74 75 76 77 78 79 7a 77 78 76 76 76 76 76 76 76 77	a b c c d e f g h i j j k l m n o o p q r s t u v w w x y z {
Ih lw1 lw lbu lhu lhu sb sh sw swr cache ll lwc1 lwc2 pref ldc1 ldc2 sc swc1	addu sub subu subu subu subu subu subu su	cvt.df cvt.wf c.ff c.urf c.eqf c.ueff c.ultf c.ultf c.sif c.nglef c.secf c.nglf	10 10 10 10 10 10 10 10 10 10 10 10 10 1	0001 0010 0011 0100 0101 0110 1001 1011 1100 1101 1110 0000 0011 0100 0110 0111 1100 1101 0110 0111 0110 0111	33 34 35 36 37 38 39 40 41 42 43 44 45 50 50 51 52 53 54 55 57 58 59	21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 30 31 32 26 33 34 35 36 37 38 39 3a 3b	# \$ \$ % & & ; () ** + + , ** / / 0 1 2 2 3 4 4 5 6 6 7 7 8 9 9 : ; ;	97 98 99 100 101 102 103 104 105 106 107 108 110 111 112 113 114 115 116 117 118 119 120 121 121	61 62 63 63 64 65 66 66 67 70 71 72 73 74 75 76 77 78 79 72 73 74 75 75 75 75 75 75 75 75 75 75 75 75 75	a b c c d e e f g h i j j k l m n o o p q r r s t u v w w x y z

(1) opcode(31:26) == 0(2) opcode(31:26) == $17_{\text{ten}} (11_{\text{hex}})$; if fmt(25:21)== $16_{\text{ten}} (10_{\text{hex}}) f = s$ (single); if fint(25:21)== $17_{\text{ten}} (11_{\text{hex}}) f = d$ (double)

IEEE 754 FLOATING-POINT STANDARD

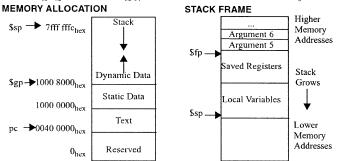
 $(-1)^S \times (1 + Fraction) \times 2^{(Exponent - Bias)}$ where Single Precision Bias = 127,

Double Precision Bias = 1023. IEEE Single Precision and



4





DATA ALIGNMENT

Double Word								
	Wo	rd		Word				
Halfword Halfword			Hal	fword	Halfword			
Byte Byte		Byte	Byte	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

Interrupt Exception Mask Code Pending M LE Interrupt

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

CLFIN	J. 4 CC	DEG			
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 (10x for Disk, Communication; 2x for Memory)

	, , , , , , , , , , , , , , , , , , , ,							
	PRE-		PRE-		PRE-		PRE-	
SIZE	FIX	SIZE	FIX	SIZE	FIX	SIZE	FIX	
$10^3, 2^{10}$	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.

```
find clock rate for fz that reduces its execution time to aft, also no instruction
 performance = clockrate / CAI
 No Cocles = time x clockrate
                                                                           to reduce to a fl
                                    \rho) Time = 7s \frac{2.5}{3.57}
                                    P2 7ime = 103 70
P2 Clock = 2.5gh2 . 30x109
 time = no instructions X CPI
                                                                            A 106 X.1 pico
B 106 X.2 pico
c 106 X.2 mano
c 106 X.2 milio
      clock rate.
                                     p2 instruction 5=30×109 2
No instructions = \frac{\text{no cycles}}{\text{CPF}}
                                 Given a program with 10° instructions divided as follows.
                                      2.5 ghz 10% 8 50% 20%
clock rate = no instr. x < pI
time
                                                                               mate all the same
                                                                               ble in yourses.
                                      cp2 1 2 3 3
                                                                               parcycle = I = no instr.
timex clarkak.
                                                                             Global CPI time vibeliate

no 1 x 10 4 x 10 5 x 10 5 x 10 9

10.4 x 10 4 x 3.5 x 10 9
                                    Time A: 1x10 x 1 / 25x109 = .4 x10
                                    Time B = 2 x 10 x 2 / 2 3 x 2 3 = 1.6 x 2
                                    Time C = 5 x 10 5 x 3 / 12 5x 10 = 6.0 x 10 4
Time new so frew = folk
                                    TimeD = 2 x10 x3/ 210 = 2,4 x5"
 Time old
                                                                           find clock cycles required?
 CPI = time x clockrate
                                 what is execution time of program an
                                                                           add up CPI x ne instanctions for
         no instructions,
                                   2 ghc machine?.
                                                                            the program.
                                                             650× 1
Execution Time = no instructions
                                  Arith legite 650 instr
                                                             100 × 5
                                                                     =2 115×10-6
                                                                                  Cool CPI for Same
                                  Store Scaries 100 "
                                                             600 X S
                                                                                  41010m2,
                                  look scores 600 "
                                                              50x2+=2,185r5
 capacitic load = Power x clark
                                                                                    (2,725×10-6)×(2.0×10°
                                  Branch Zeydis 50 "
                                                                                     total instructions (1458,
                                  Geometric Mean Moch Pale Pata: Freet, Find all
- increasing clock cycle time
  lames clock rate,
                                   the ratios. Take next gen /old gen. With a,
                                                                                     Largest Adative
                                    all the ratios, and raise that to invise
- Spendap = time old = %
                                                                                        (Mange
                                    of court of latins calculated. If 7 ratios,

(sum of ratios). | find avenue cp
Andehis Law
                                                                                       largest rost o
                                                                                        calculated.
                                                                   find avenue CPI
 Speedup = old
                                 Largest relative Change in voltage Arith 70% Zeyles
     unmosited to modified
                                                                   Loadstue 10% 6 cycles
Cycle Time = 1 lhz = 1 cycle
rate per second.
                                    younge?
                                                                  Branch 20°lo
                                                                                     3 cycles
16 bit single point Floating Point
                                   Floating Point Addition
                                                                     2 \times .7 + 6 \times .1 + 3 \times .2 = 2.6
19.03125 = 10011.00001
                                     make Exponents same.
 1.00/100001 ×10 4=€-15
                                     add values up, use 28 comp
                                                                    19,63125 to
                                                                                      precision floating of
                                     if necessary. Dfop lasteams
                                                                     19/2 R=1
                                                                                      .03125 x2
                                      bit, itans,
c 10011 0011 000010
                                                                     912 R=1
                                                                                      .0625 x2
                      Shiffed off
                                                                                      125 x 2
Guard > Abund + sticky bits.
                                                                     41 2 1 2=0
                                    floating point multiplication
                                                                                      . 25 X2
                                                                     2/2 1=0
                                  - add exponents.
                                                                                      15 KZ
                                                                     1/2 1=1
                                    conallyly values.
   10 0 1
                                                                     o/L done
                                                                                       00001
                                    - count number of digits after
                                                                     1001)
                                     first . by that amount of 10011.00001 = 1.001100001 x 24
                                     . For both values. Shift
                                                                  4= E-127
                                   - check sign band on originals. E = 131
```

non pipelined arch running at 2.56hz 3.75 × 29.625 in Floating Point 17-8 conditional that takes scycles to finish an instruction, 3.75 = 11.11 = 1.111 ×2 / 7= 0111 -8=1600 you want 5 pipelined Stages, Now running 29.625 = 11101.101 = 1.1101/01 × 2" +1 0 0 0 at 26hz . memory 30/o of instructions, stall of so 1.1101101 (10,01) (10,01) (10,01) (10,01) Cyclis happens 2°10 · branch 20% of instructions, stall of 2, (100,011) 0.11101101 0.011101101 cycles, happens 70% 0.0011101101 what is speedup? 11.0111100011 (10000,01111) Speed = 5 2.5 1.10111100011 x2 since addition, no carry in so solution is 4+1+1 = 6 6= E-127 E=133 1+ 134.02×50+12×12 × 2 hs 6hz no pipeline, instruction buffers Speedup = $\frac{3}{1+.3+.08} \times \frac{2}{2.5}$ -3x -6 by booths that negationstruction load penalties X=-3=1101 Y=-6=1010 -y = 6 = 0110) 80% of the time. /32kb cleache, las access, 4 way associatie, with though, 2% miss rate for U U X X-1 0000 0000 1101 0 data and 1/60 for instructions, RAW - read after write 0110 0000 1101 6 Ing 12 cooks, 12 ns acces, 4 way bestoriche, when a read instruction occurs before a previous write instruction has finished. write back, write-already 30% mas late, 1101 0000 1110 1 1110 1000 0111 0 20% dity. WAR-write after read 0100 1000 0111 0 166 ran 5003 accisi 0010 0100 1011 when a write instruction occurs before a previous 0001 0010 1161 1 / Branch delay slot! Read instruction has finished, 00010610=18 No , a branch dalay 5101 is only useful to what is the effective reduce the ponalty of address jamps in a WAW - write after write access fine for lding pipelined, machine, In a pipelined machine, when a write instruction who the branch occurs, two or three occurs befor a previous justinutions2. write instruction has finished. instructions following the branch are topically already looks, These instruction [At loads = MR buffer (TC: +MRL: (TL2+MRL2 (1+dirty)] Tram)) must be firshed, rodrieny the efficiency = ,2(1+1/60 (12++3(1+,2)50)) of the pipeline. The branch delay = ,3~5 reduces the penalty by one. CPI Penalty Recommended Changing to make 1d. h R2, 0(R3) Penaltycp1 = Eat londs x CK fusteradd R3, R3,4 2 Warwithl = .3ns ×1.56h2 Two thirds of the penalty is mul R2, R2, R2 3 Raw with 1 due to the access time of - MG cycles St. b 12,0(23) 4 level one cache. I would look Raw with 2,3 at trying to reduce or sub Rz, R3, R4 5 Raw with 2, war 4 negate this (irst 1 bnez RZ, square 6 Raw with 5

f,6,h,1,J \$150 to 54 1, add 9 to, \$156, \$50 tempo=A+f Z. ald At1, \$17, \$31 templ + B+9 A=\$56 3. IN \$0, 0(\$10) B-557 F= ACF] 4. adl: 5+2, 51to, 4 temp2 =temp0+1 5. Iw 9to, 0(8+2) tempo = \$ tempz t= AEt) 6. ald \$to, \$to, \$50 tempo = tempo + F TO = FA ACFFI] 7. Sw \$1+z, 0(\$1) B (6) = 4 (c+) temp2= * temp)

\$156=16 f=8

- Block offset sice = 2^{1/5}, 32K 2 cam try 25 bits
- $\frac{10}{2} 2^{15} = 2^{25} \text{ pages } \times 64 \text{ words} = 2^{25} \times 2^{6} = 2^{3} \text{ pages}$
 - O yes, because pages arent written back to disk right away.

 ses, need valid to determine if entry in page is valid
 - O cache registers should contain actual additisses so no translation cache is necessary to context from witned to physical, cache should map to physical memory
 - (e) Setup a TLB, transition look aside butter so that you will have part of the page table locally, and wouldn't need to go to memory to get page table into,