1.5 Exerc.

Q. 1.7 [20].

	A	B	C	D
P1 CPI	1	2	3	3
P2 CPI	2	2	2	2
	0.1	0.2	0.5	0.2

$\#I = 1E6 = 10^6$

$$\text{Ex. Time} = \#I \times \frac{\overbrace{\frac{C}{I}}^{CPI}}{} \times \frac{Sec}{C}$$

$2.5\,GHz = 2.5\,E10^9 \frac{cycles}{sec}$

$3\,GHz = 3\,E10^9 \frac{cycles}{sec}$

- $P_{1_{time}} = \left[\left((0.1)1 + (0.2).2 + (0.5)3 + (0.2)3\right).10^6\right] \frac{1}{(2.5).10^9}$

$$= \boxed{10.4 \times 10^{-4}}$$

$$P_2 = \left[(0,1).2 + (0.2)2 + (0.5)2 + (0.2)2)10^6\right] \cdot \frac{1}{3 \times 10^9}$$

$$= \boxed{6.66 \times 10^{-4}} \rightarrow \text{Faster}$$

Part b)

Global CPI

$$CPI \Rightarrow \frac{I \times CPI}{\text{Clock Rate}} = \text{Time}$$

$$\Rightarrow CPI = \frac{\text{Time} \times \text{Clock Rate}}{I} \rightarrow$$

$$CPI_{P_1} = \frac{10.4 \times 10^{-4} \times 2.5 \times 10^9}{10^6} = 2.6$$

$$CPI_{P_2} = \frac{6.66 \times 10^{-4} \times 3 \times 10^9}{10^6} = \boxed{2.00} \leftarrow \text{faster}$$

$\#\,cycles = \#I \times CPI$

$P_1 = (10^6 \times 0.1 \times 1) + (10^6 \times 0.2 \times 2) + (10^6 \times 0.5 \times 3) + (10^6 \times 0.2 \times 3) = \boxed{26 \times 10^5} \rightarrow$ number of cycle of P_1

$P_2 = 10^6 \times 2 = \boxed{20 \times 10^5} \rightarrow$ # of cyc. of P_2

1.13.4

	Clock Rate	CPI	#I
P_1 →	4 GHz	0.9	$5 \cdot 10^9$
P_2 →	3 GHz	0.75	$1 \cdot 10^9$

from Question

40% floating p.

0.4

$$MFLOPS = \frac{\#FPI \times 10^{-6}}{Time}$$

$$P_{1\,Time} = \frac{\#I \times CPI}{ClockRate} = \frac{5 \times 10^9 \times 0,9}{4 \cdot 10^9} = 1.125 \text{ sec}$$

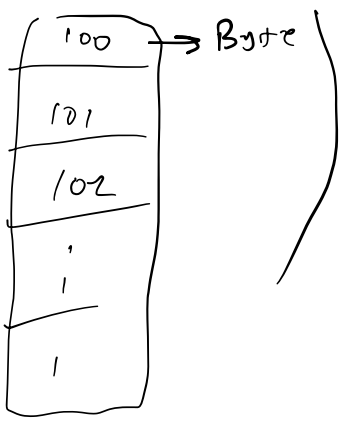
$$P_{2\,Time} = \text{"} = \frac{1 \times 10^9 \times 0,75}{3 \times 10^9} = 0.25 \text{ sec}$$ → faster

$$MFLOPS_{P_1} = \frac{5 \times 10^9 \times (0,4) \times 10^{-6}}{1,125} = 1,78 \times 10^3$$

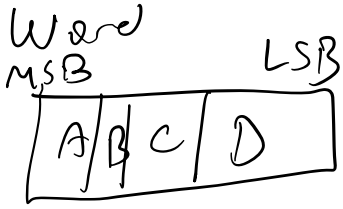
$$MFLOPS_{P_2} = \frac{1 \times 10^9 \times 0,4 \times 10^{-6}}{0.25} = 1,6 \times 10^3$$

CHP2

add rd, rs, rt



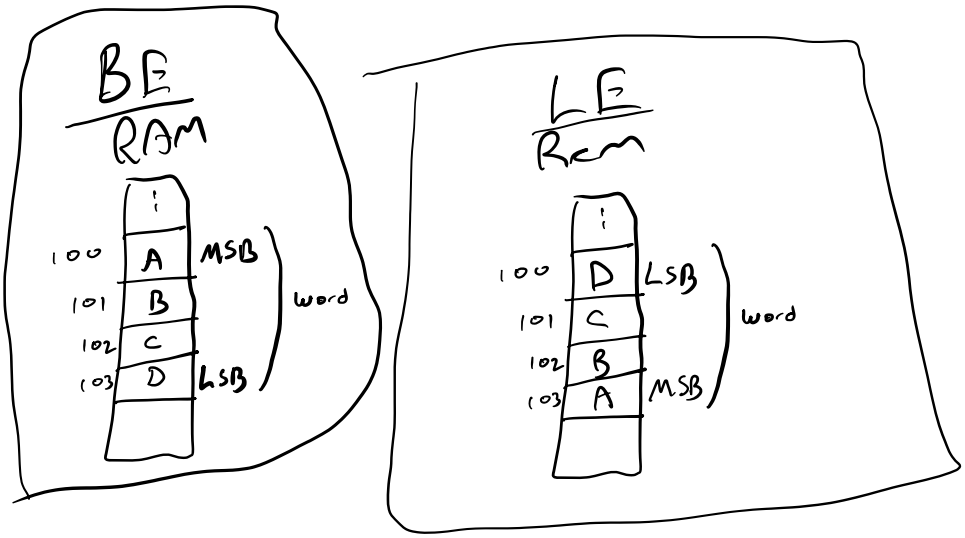
32 bit = Word

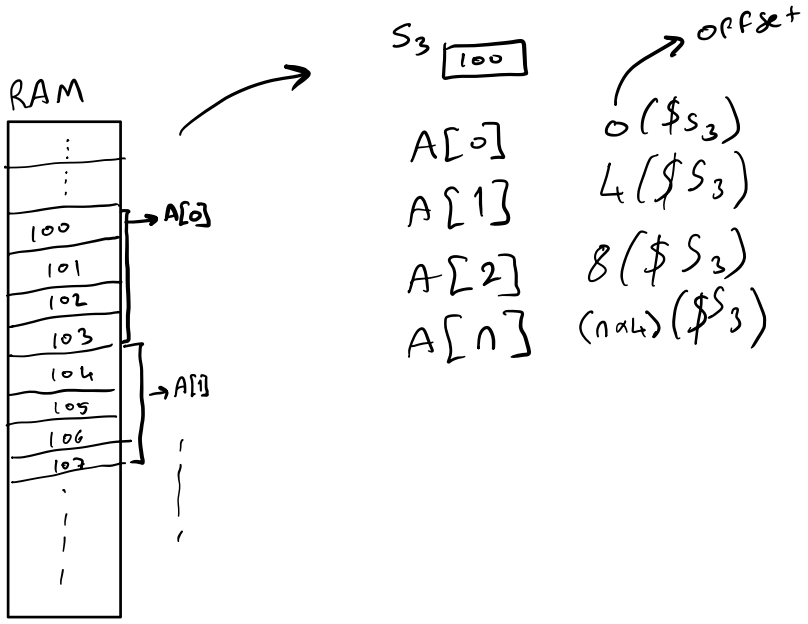


Big Endian (MIPS) = BE

↓

MSB goes to lower addr




RAM
100
101
102
103
104
105
106
107
A[0]
A[1]
S_3
100
offset
A[0]
A[1]
A[2]
A[n]
0(S_3)
4(S_3)
8(S_3)
(n*4)(S_3)

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

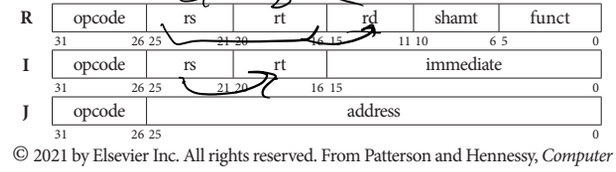
①

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

(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

BASIC INSTRUCTION FORMATS



ARITHMETIC CORE INSTRUCTION SET

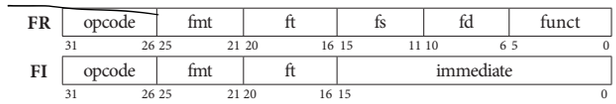
②

NAME, MNEMONIC		FOR-MAT	OPERATION		OPCODE / FMT / FT / FUNCT (Hex)
Branch On FP True	bc1t	FI	if(FPcond)PC=PC+4+BranchAddr	(4)	11/8/1/--
Branch On FP False	bc1f	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 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] op F[ft]) ? 1 : 0		11/10/--/*y*
FP Compare Double	c.x.d*	FR	FPcond = ({F[fs],F[fs+1]} *op* {F[ft],F[ft+1]}) ? 1 : 0		11/11/--/*y*

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

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]+SignExtImm]	(2)	31/--/--/--
Load FP Double	ldc1	I	F[rt]=M[R[rs]+SignExtImm]; F[rt+1]=M[R[rs]+SignExtImm+4]	(2)	35/--/--/--
Move From Hi	mfhi	R	R[rd] = Hi		0/--/--/10
Move From Lo	mflo	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] >> shamt		0/--/--/3
Store FP Single	swc1	I	M[R[rs]+SignExtImm] = F[rt]	(2)	39/--/--/--
Store FP Double	sdc1	I	M[R[rs]+SignExtImm] = F[rt]; M[R[rs]+SignExtImm+4] = F[rt+1]	(2)	3d/--/--/--

FLOATING-POINT INSTRUCTION FORMATS



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
$k0-$kl	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

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
siti	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	la	SUB	90	5a	Z
	divu		01 1011	27	lb	ESC	91	5b	[
			01 1100	28	lc	*FS*	92	5c	\
			01 1101	29	1d	GS	93	5d	]
			01 1110	30	le	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		100011	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	2*f*	/	111	6*f*	0
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
idc1		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	3*f*	?	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

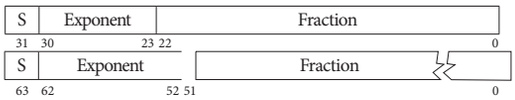
④

IEEE 754 Symbols

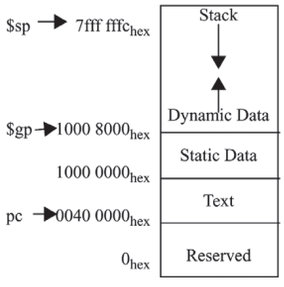
Exponent	Fraction	Object
0	0	± 0
0	≠ 0	± Denorm
1 to MAX - 1	anything	± F1. Pt. Num.
MAX	0	± ∞
MAX	≠ 0	NaN

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

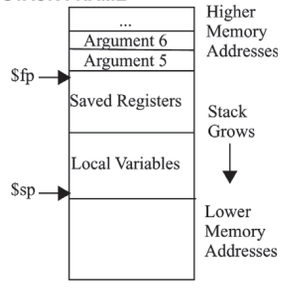
IEEE Single Precision and Double Precision Formats:



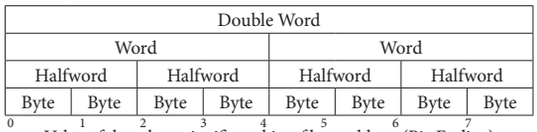
MEMORY ALLOCATION



STACK FRAME

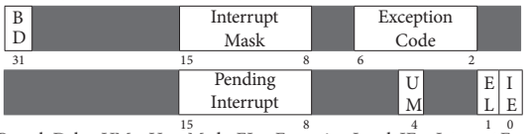


DATA ALIGNMENT



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

SIZE	PREFIX	SYMBOL	SIZE	PREFIX	SYMBOL
1000^1	Kilo-	K	2^{10}	Kibi-	Ki
1000^2	Mega-	M	2^{20}	Mebi-	Mi
1000^3	Giga-	G	2^{30}	Gibi-	Gi
1000^4	Tera-	T	2^{40}	Tebi-	Ti
1000^5	Peta-	P	2^{50}	Pebi-	Pi

SIZE	PREFIX	SYMBOL	SIZE	PREFIX	SYMBOL
1000^6	Exa-	E	2^{60}	Exbi-	Ei
1000^7	Zetta-	Z	2^{70}	Zebi-	Zi
1000^8	Yotta-	Y	2^{80}	Yobi-	Yi
1000^9	Ronna-	R	2^{90}	Robi-	Ri
1000^{10}	Quecca-	Q	2^{100}	Quebi-	Qi

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

i = 3 0011 x4 = 12

1100

BD 2 1

Let i = 2 = 0010

Ex 2.7

~~100~~ A = 4 Byte!

B[8] = A[i] + A[J]

leah = 16
8ah = 32

i = $S3
J = $S4

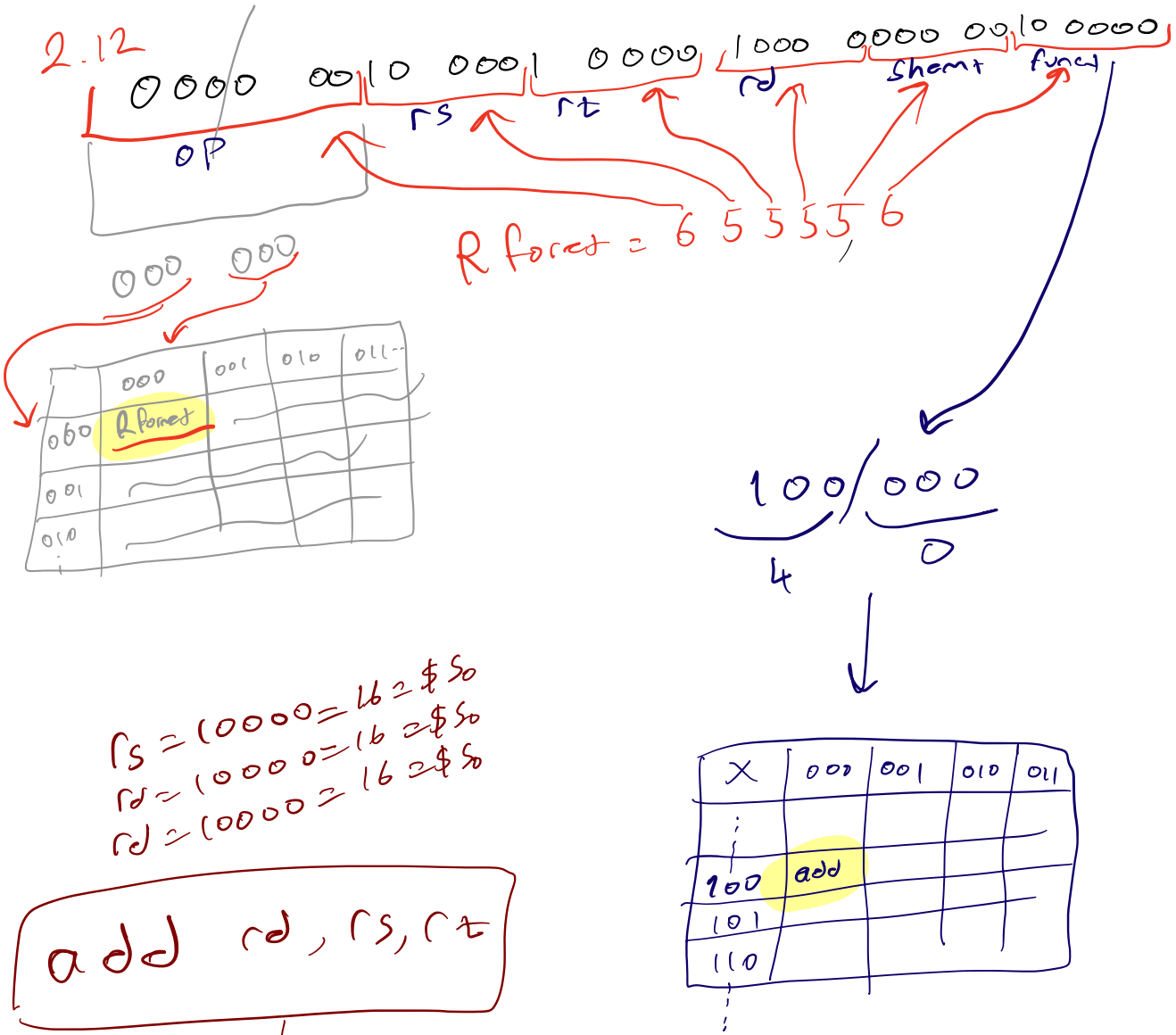
A = $S6 → 16($S6)
B = $S7 → 32($S7)

SLL $t0, $S3, 2 → 4 x i = 16
add $t0, $t0, $S6 → t0 add A[i] / Base
S6 = (Base + 16)
Lw $t0, 0($t0) → t0 → A[J]

shift 2 times

Sll $t1, $S4, 2
add $t1, $t1, $S7
Lw $t1, 0($t1)

add $t0, $t0, $t1
sw $t0, 32($S7) → 8. index of B


2.12
0000 0010 0001 0000 1000 0000 0010 0000
op
rs
rt
rd
shamt
funct
Rformat = 6 5 5 5 5 6
000 000
000
001
010
011
100
101
110
Rformat
100 000
4
0
X
000
001
010
011
100
add
101
110
rs = 10000 = 16 = $s0
rd = 10000 = 16 = $s0
rd = 10000 = 16 = $s0
add rd, rs, rt
add $s0, $s0, $s0
in C → a = a + a;

2 13

sw $t1, 32($t2)

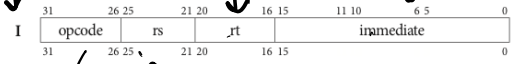
$t0 = 8

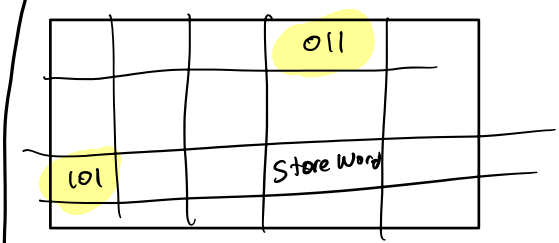
$t1 = 9

$t2 = 10

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
$k0-$kl	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

Store Word	sw	I	M[R[rs]+SignExtImm] = R[rt]	(2) 2b$_{hex}$





$rt = t1 = 9 = 01001$

$rs = t2 = 10 = 01010$

$101011 = opcode \rightarrow (101011)_2 = (2B)_{16}$

2 11 = B

32 = 100000

but we need 16bit

0000 0000 1000000

101011 01010 01001 0000 0000 0010 0000

A D 4 9 0 0 2 0

$(2B5260)_{16}$ //

JUMP

main()

func() (...)

Assembly

L:

JUMP

Jal L

J L

---

slt rd, rs, rt → if(rs < rt){ rd = 1 }
else { rd = 0 }

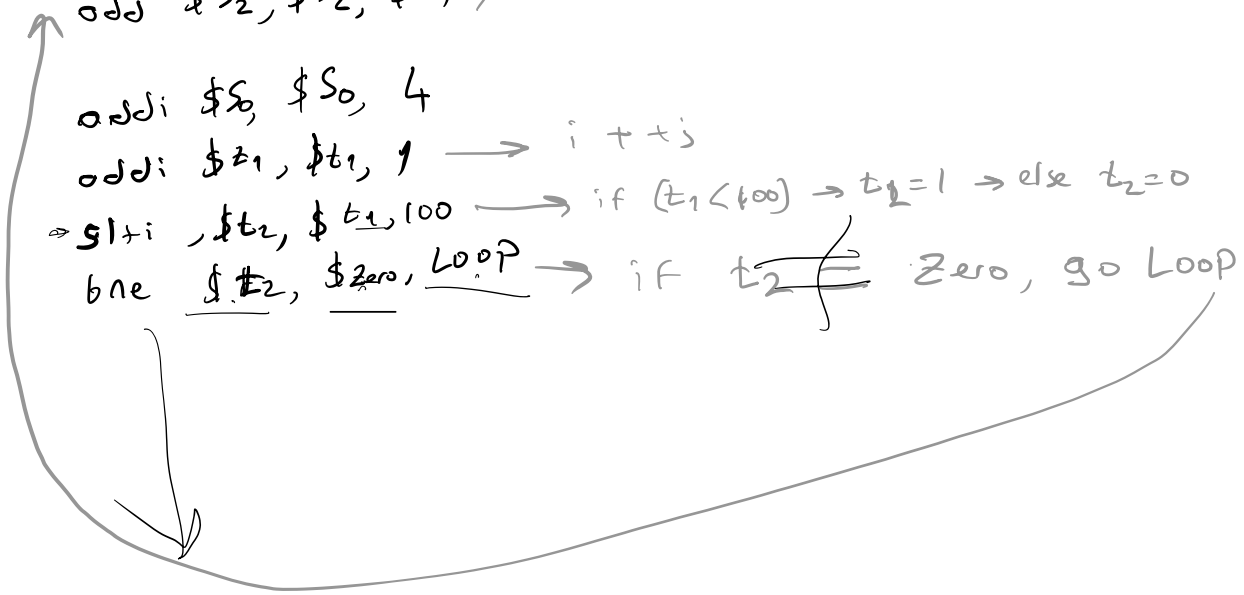
slti rt, rs, constant → if (rs < constant){ rt = 1 }
else { rt = 0 }

2.27)

i = $t1

result = $s2

```
add  $t1, $zero, 0      → i=0;
Loop: lw  $s1, 0($s0)   → result += MemArray[i]
      add $s2, $s2, $s1
      addi $s0, $s0, 4
      addi $t1, $t1, 1  → i++;
      slti $t2, $t1, 100 → if (t1<100) → t1=1 → else t2=0
      bne  $t2, $zero, Loop → if t2 == zero, go Loop
```



i = 0

result = 0

```
for (i=0; i<100; i++){
    result += MemArray[i];
}
```

3.9

151 → 10010111 → (-1 / flip bits) → 01101001 = (-105)

neg.

214 = → 11010110 → 0010 1010 = -42

neg

-147

3.25

63.25

.25 × 2 = 0.50

0.5 × 2 = 1

01

111111.01

1 1
2 1
3 1
8 1
16 1
32 1

111111.01 = 1.1111101 × 2^5

fraction

5 times

5 + 127 = 132 = 10000100

0 10000100 11111010000000000000000

sign | 8 bit exponent | 23 bit

16 bit

5

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:

S	Exponent	Fraction

31 30 23 22 0

S	Exponent	Fraction

63 62 52 51 0

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