北京大学信息科学技术学院考试试卷

考试科目： 计算机组成 **姓名：** _____ **学号：** _____

考试时间： 2016 年 4 月 20 日 **任课教师：** 陆俊林

题号	一	二	三	四	五	六	七	八	**总分**
分数									
阅卷人									

北京大学考场纪律

1、考生进入考场后，按照监考老师安排隔位就座，将学生证放在桌面上。无学生证者不能参加考试；迟到超过 15 分钟不得入场。在考试开始 30 分钟后方可交卷出场。

2、除必要的文具和主考教师允许的工具书、参考书、计算器以外，其它所有物品（包括空白纸张、手机、或有存储、编程、查询功能的电子用品等）不得带入座位，已经带入考场的必须放在监考人员指定的位置。

3、考试使用的试题、答卷、草稿纸由监考人员统一发放，考试结束时收回，一律不准带出考场。若有试题印制问题请向监考教师提出，不得向其他考生询问。提前答完试卷，应举手示意请监考人员收卷后方可离开；交卷后不得在考场内逗留或在附近高声交谈。未交卷擅自离开考场，不得重新进入考场答卷。考试结束时间到，考生立即停止答卷，在座位上等待监考人员收卷清点后，方可离场。

4、考生要严格遵守考场规则，在规定时间内独立完成答卷。不准交头接耳，不准偷看、夹带、抄袭或者有意让他人抄袭答题内容，不准接传答案或者试卷等。凡有违纪作弊者，一经发现，当场取消其考试资格，并根据《北京大学本科考试工作与学术规范条例》及相关规定严肃处理。

5、考生须确认自己填写的个人信息真实、准确，并承担信息填写错误带来的一切责任与后果。

学校倡议所有考生以北京大学学生的荣誉与诚信答卷，共同维护北京大学的学术声誉。

以下为试题和答题纸，共 12 页。

装订线内

不要答题

得分	

一、单选题（每小题 2 分，共 20 分）

1. 以下人物中，没有参与电子计算机 ENIAC 研发工作的是（　　）

   A. 约翰·莫克利　　B. 约翰·埃克特

   C. 约翰·阿塔纳索夫　　D. 约翰·冯·诺依曼

2. 通常认为第一台存储程序式的电子计算机是（　　）

   A. EDSAC　　B. EDVAC　　C. ENIAC　　D. UNIVAC

3. 通常情况下，以下关于不同类型计算机的相互比较，哪句是正确的（　　）

   A. 超级计算机的运算性能是最高的

   B. 大型计算机的体积是最大的

   C. 小型计算机的功耗是最小的

   D. 微型计算机的制造成本是最高的

4. 以下不属于冯·诺依曼计算机结构五个基本组成部分的是（　　）

   A. 控制器　　B. 存储器　　C. 运算器　　D. 计时器

5. 按两个 32 位源操作数位置划分，MIPS 和 x86 的加法指令都能够支持的是（　　）

   A. 寄存器+寄存器　　B. 寄存器+存储器

   C. 寄存器+立即数　　D. 存储器+立即数

6. 关于 x86 指令构成，以下说法错误的是（　　）

   A. 指令中可以只包含操作码　　B. 指令中可以只包含操作数

   C. 指令长度是可变的　　D. 操作数在操作码之后

7. MIPS 算术运算指令会影响的标志位是（　　）

   A. 溢出标志　　B. 零标志　　C. 方向标志　　D. 以上都不对

8. MIPS 指令中指示目的寄存器编号的位域是（　　）

   A. rd 或 rt　　B. rd　　C. rs 或 rd　　D. rt

9. 对于带符号数的除法运算，-9÷4 的商和余数分别是（　　）

   A. -2 和-1　　B. -2 和 1　　C. -3 和 3　　D. 3 和-3

10. 以下关于返回地址栈 RAS 的说法，正确的是（　　）

    A. RAS 中保存了 RET 指令的地址

    B. RAS 是一个“先进先出”的结构

    C. 访问 RAS 要使用栈寄存器 SP

    D. 执行 CALL 指令时会操作 RAS

得分

第二题（20 分）

分析如下 MIPS 机器代码，按照各小题指导的步骤，完成手工反汇编。

Address	**Instruction**
0x00804000	0x00001025
...	0x00441020
	0x20A5FFFF
	0x0005402A
	0x11000001
	0x08201001

1、根据上述代码填写下表，包括每条指令的类型和编码。需要根据 MIPS 指令编码格式的规则切分“other fields”域，并用明显的竖线隔开。指令编码采用十进制格式填写（J 型指令的地址域采用十六进制填写）。

Address	**R/I/J**	**opcode**	**other fields**
0x00804000			
...			

2、将上表中的指令编码转换成汇编助记符，并补全第一列的地址。寄存器编号用寄存器名称（如$t0），分支指令的目标地址用 Label 的形式表示，并根据需要填写在 Label 栏中相应位置。

Address	Label	Instruction
0x00804000		

3、将上述汇编语言代码还原成最有可能的 C 语言代码，涉及的变量依次用 a、b、c 等标记，需要指明变量与寄存器之间的对应关系。然后用文字说明这段代码可能的用途，并指出其中可能存在的问题。

答：

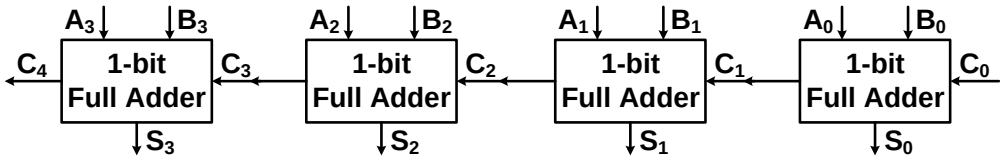
得分

第三题（15 分）

1. 请以 PMOS 和 NMOS 为基本单元，画出一个与门的结构图。

2. 请以逻辑门为基本单元，画出 1 位全加器的门电路实现结构。

3. 下图是 4 个全加器组成的 4 位行波进位加法器，请结合第 1 小题，给出其关键路径的门延迟数，并说明计算方法，然后推导出 n 位行波进位加法器的门延迟通用表达式。



答：

4. 根据第 2 小题图，给出第 i 个全加器的进位输出 C_{i+1} 的产生逻辑表达式。
答：

5. 如果采用“超前进位”的方式，则请写出 C_4 的产生逻辑表达式。
答：

6. 请画出 4 位超前进位加法器的关键路径（延迟最长的电路），并标出其中重要的信号，如 A_0、B_1、C_2、S_3 等。
答：

得分

第四题（15 分）

某假想指令系统有如下四条指令（R 为寄存器编号，M 和 L 为存储器地址）：

- ADD R, M //功能：将 R 的内容与 M 中的内容相加后存入 R
- LOAD R, M //功能：将 M 中的内容装入 R
- STORE M, R //功能：将 R 的内容存入 M 中
- JMP L //功能：无条件转向 L 处

假设对应的控制器类型为微程序控制器，其微指令格式如下所示：

PC_{OUT}	MAR_{IN}	M 读	PC 加	MDR_{OUT}	IR_{IN}	IR_{OUT}	PC_{IN}
$R1_{IN}$	Y_{IN}	$R1_{OUT}$	MDR_{IN}	M 写	add	Z_{OUT}	End

1、请补全下表中的微程序代码和其它相关内容：

微地址	指令	机器周期	微操作字段	下址字段
0000	取指令	T1	1111 0000 0000 0000	0 0001
0001		T2		
0010	LOAD(00)	T3		
0011		T4		
0110	ADD(01)	T3	0110 0010 0000 0000	0 0100
0100		T4	0000 1000 0100 0000	0 0101
0101		T5		0 0111
0111		T6	0000 0000 1000 0010	
1010	STORE(10)	T3		0 1000
1000		T4	0000 0000 0011 1000	0 1001
1110	JMP(11)	T3		0 1001
1001	End	Tn		

2、如果执行如下代码，请写出控制器发出的微地址序列（如：0000→0001→……）

程序代码	微地址序列
LOAD R1, MEM_A	_____
LAB_1: ADD R1, MEM_B	_____
JMP LAB_3	_____
ADD R1, MEM_C	_____
ADD R1, MEM_D	_____
LAB_3: STORE MEM_Z, R1	_____

3、请分析硬布线控制器和微程序控制器各自的优点和缺点。

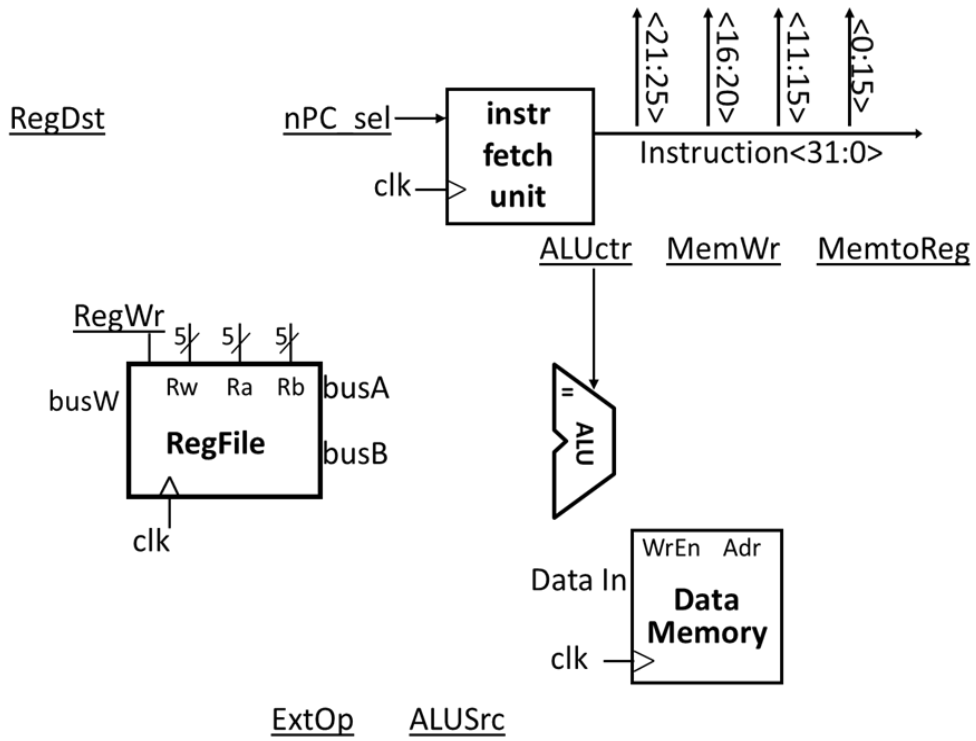
	硬布线控制器	微程序控制器
优点		
缺点		

得分	

第五题（15 分）

下图为单周期 MIPS 处理器的数据通路结构图，请补全其中缺失的功能部件和连线，具体要求包括：

- 功能部件上须明确标明其功能。例如，若新增一个多选器，则在画出多选器的图形后，须用文字标明“多选器”或“MUX”，并用 0、1 等标出每个输入信号的选择编号。
- 宽度超过 1 位的连线上须标明其位宽，形式参见下图中 RegFile 的 Rw、Ra 和 Rb 所接的信号线。
- 图中带下划线的是控制信号的名称，须将其与对应的功能部件相连。



得分

第六题（15 分）

某个经过简化的假想计算机系统中，CPU 的转移预测部件采用了 BTB 结构，在取指令的同时检索 BTB，BTB 未命中时预测不转移。该 BTB 共 4 个表项，采用直接映射方式分配表项。其历史位采用与 Pentium 相同的两位历史信息转移预测器。当前系统正在运行中，CPU 将要执行的程序段如下所示。为简化计算，假定所有指令长度均为 1 个字节，且被省略的代码中没有转移类指令。

```
            JMP   L_start          ;指令地址：42011H
            …
L_start:    MOV   CX, [BX]         ;指令地址：42030H；设[BX]=8
            XOR   DX, DX           ;指令地址：42031H
L_loop1:    MOV   AX, [BX+8]       ;指令地址：42032H；设[BX+8]=6
            CMP   AX, CX           ;指令地址：42033H
            JZ    L_exit           ;指令地址：42034H
L_loop2:    INC   DX               ;指令地址：42035H
            …
            DEC   AX               ;指令地址：4204EH
            JNZ   L_loop2          ;指令地址：4204FH
            …
            LOOP  L_loop1          ;指令地址：4206AH
L_exit:     MOV [BX+16],DX         ;指令地址：4206BH
```

当前 BTB 的内容				
转移目标地址	转移指令地址	历史位	有效位	
4206BH	42034H	01	1	表项 0
42030H	42011H	10	0	表项 1
42032H	4206AH	01	1	表项 2
42032H	4004FH	01	1	表项 3

（1）请描述第一次执行 JMP 指令时，BTB 的所有操作（是否命中以及命中哪个表项、是否预测转移、预测是否正确、是否修改表项以及修改哪个表项，下同）
答：

（2）请根据第一次 JMP 指令执行完成后的 BTB 内容填写下表，仅填写相比初始表有改动的行，内容不变的行保持为空即可。

BTB				
转移目标地址	转移指令地址	历史位	有效位	
				表项 0
				表项 1
				表项 2
				表项 3

（3）请描述第一次执行 JZ 指令时，BTB 的所有操作
答：

（4）请根据第一次 JZ 指令执行完成后的 BTB 内容填写下表，仅填写相比前一张表有改动的行，内容不变的行保持为空即可。

BTB				
转移目标地址	转移指令地址	历史位	有效位	
				表项 0
				表项 1
				表项 2
				表项 3

（5）请描述第一次执行 LOOP 指令时，BTB 的所有操作

答：

（6）请根据第一次 LOOP 指令执行完成后的 BTB 内容填写下表，仅填写相比前一张表有改动的行，内容不变的行保持为空即可。

BTB				
转移目标地址	转移指令地址	历史位	有效位	
				表项 0
				表项 1
				表项 2
				表项 3

（7）在第一次执行完成 LOOP 指令后，到执行标号 L_exit 处的指令时，总共执行了_____次转移指令，其中 BTB 命中_____次，预测正确_____次。

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

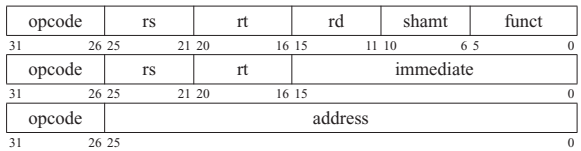
①

CORE INSTRUCTION SET

NAME, MNEMONIC		FORMAT	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

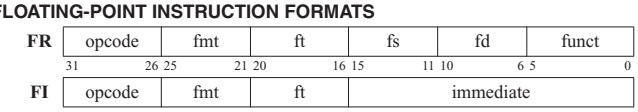
②

NAME, MNEMONIC		FORMAT	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
$k0-$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

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.

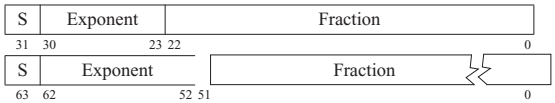
④

IEEE 754 Symbols

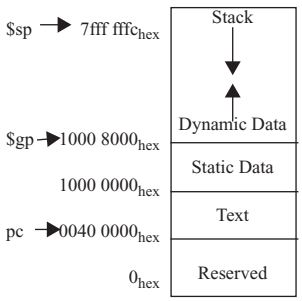
Exponent	Fraction	Object
0	0	± 0
0	≠0	± Denorm
1 to MAX - 1	anything	± Fl. Pt. Num.
MAX	0	±∞
MAX	≠0	NaN

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

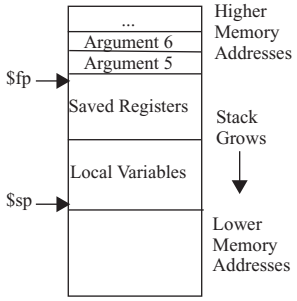
IEEE Single Precision and Double Precision Formats:



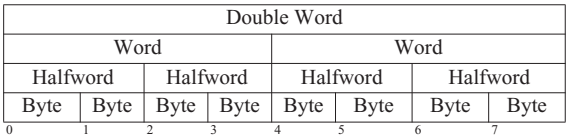
MEMORY ALLOCATION



STACK FRAME

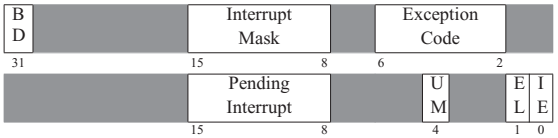


DATA ALIGNMENT



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