Computer Architecture

1 ISA (Instruction Set Architecture) (10%)

ISA	Number of General registers	Number of FP registers
Intel X86 (80386)	8 (eax to esp)	8 (ST0 to ST7)
Intel X86-64	16	32 ZMM registers
Intel Itanium	128	128
ARMv7 (VPFv2)	16	16
ARMv8	31	32

The above table shows the number of registers provided by Intel x86, Intel/HP Itanium, and ARM ISA (Instruction Set Architecture).

A. (1 point) Please list one or more ISAs that provide **no** registers at all.

B. (5 points) Registers are usually used to hold temporaries. Modern compilers also use registers to hold frequently used local variables, pointers, and so on. Ideally, the more the registers, the more objects could be allocated to registers and memory references to those objects could be saved. The trend of ISA has clearly shown that new architectures tend to have more registers.
So is it a good idea to take a quantum leap to go for 1024 registers in an ISA. You must argue for your answer, just say yes or no will get no credits.

C. (2 points) In each ISA's calling convention, the set of general purpose registers is usually divided into caller-save and callee-save two subsets. Which type of procedures could benefit most from such a register partition?

D. (2 points) Most ISAs have separate register files: general purpose registers, floating point registers, and SIMD (e.g. Intel SSE/ZMM and ARM Neon) registers. Why do most modern ISAs provide different set of registers? Would it be better to use just one set of general purpose registers? For example, we could have instructions such like:

```
Add   R1, R2, R3    /* integer add */
FADD  R1, R2, R3    /* Floating point add */
PADD  R1, R2, R3    /* SIMD Add */
```

Although these are different type of instructions, they could use the same general purpose registers.

見背面

③

OPCODES, BASE CONVERSION, ASCII SYMBOLS

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.

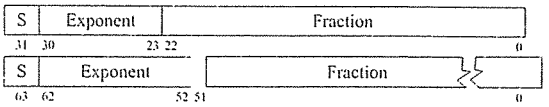
④

IEEE 754 Symbols

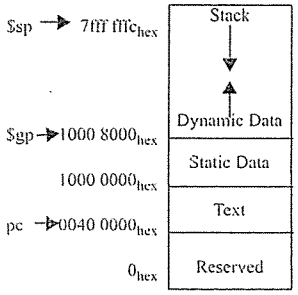
Exponent	Fraction	Object
0	0	± 0
0	≠0	± Denorm
1 to MAX - 1	anything	± Fl. Pt. Num.
MAX	0	±∞
MAX	≠0	NaN

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

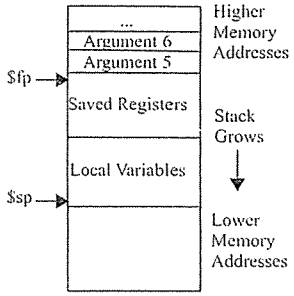
IEEE Single Precision and Double Precision Formats:



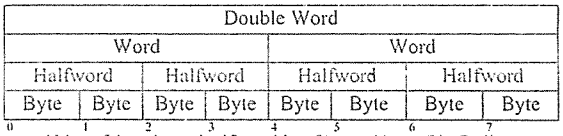
MEMORY ALLOCATION



STACK FRAME

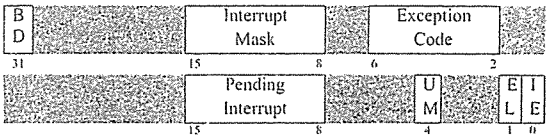


DATA ALIGNMENT



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.

Fig 1. MIPS Reference Data for Computer Architecture question 2.

接次頁

2 DATA REPRESENTATION (7%)

On a MIPS machine (Fig. 1) running UNIX, we observed the following binary string stored in memory location x.

```
0011 1111 0111 0000 0000 0000 0000 0000
```

This binary could mean many different things,

A. (1 point) If this is an integer number, what value is it?
B. (1 point) If this is a single precision floating point number, what value is it?
C. (1 point) If this is an instruction, what instruction is it?
D. (1 point) If this is a C string, what string is it?
E. (1 point) If this binary string was observed on your x86 desktop, what would be your answer for (D)?
F. (2 points) Suppose we have three variables a, b and c. Give a case where (a+b)+c computes a different value than a + (b+c) on a MIPS microprocessor.

3 BRANCH PREDICTION (8%)

For pipelined processors, control hazards could significantly decrease the performance. Dynamic branch prediction techniques have been successfully adopted in many modern processors to reduce performance penalty caused by control hazards. However, unlike direct branches (e.g. BEQ L1 in ARM or BEQ $t0,$t1, L1 in MIPS), indirect branches are usually difficult to predict.

A. (2 points) Please give at least one static and one dynamic branch prediction scheme used for **direct branches** in microprocessors.
B. (4 points) List conditions where indirect branches, instead of direct branches, are used? Which one of the listed cases is most frequent?
C. (2 points) Please explain why indirect branches are hard to predict?

4 (10%)

Cache block size is an important design parameter for cache architecture. Assume a 1-CPI (Cycle-per-Instruction) machine with an average of 1.4 memory references (both instruction and data) per instruction. Assume the CPU stalls for cache misses.

Answer the following questions using the cache miss rates for different block sizes listed in the following table.

見背面

Block Size (Bytes)	8	16	32
Miss rate	8%	4%	3%

A. (5 points) If the miss penalty is 24+ B (block size in bytes) cycles, what is the optimal cache block size? Please show how you derive the answer.
B. (5 points) If critical-word-first is implemented in the cache, what is the optimal cache block size? Please show how you derive the answer.

5 (10%)

You are given a task to parallelize the following problem in a multi-core architecture:

```
for (i = 0; i < N; i = i+1)
        for (j = 0; j < N; j = j+1) {
            r = 0;
            for (k = 0; k < N; k = k+1)
                    r = r + y[i][k]*z[k][j];
            x[i][j] = r;
        };
```

A. (5 points) Is this a weak-scaling or strong-scaling problem? Please explain your answer.
B. (5 points) Is it possible to partition this problem among cores such that there are no cache coherency misses? Please explain your answer.

6 (5%)

(5 points) In current disk storages, the operating system generally completes all I/Os asynchronously via interrupts. But recent studies show that for future NVM (Non-Volatile Memory) storage system, which has significantly lower access latency than disks, the synchronous approach (i.e., polling) could be more efficient than the interrupt approach. Please provide rationale behind this.

接次頁

Operating System

7 (15%)

Consider short-term process scheduling. Please answer the following two problems:

A. (10 points) The Round Robin scheduling algorithm (RR) lets every process run for a given time quantum and then switch to another process so as to have more fair sharing of the CPU time. Please give me your argument why a small time quantum is bad to the average turnaround time, where the turnaround time of a process is the difference between its completion time and its ready time of the process.
B. (5 points) Why having processor affinity is good to processes in using data or instruction cache.

8 (10%)

(10 points) Consider a process synchronization problem, in which we have 5 people competing for 4 chairs, and each chair can be used by one person at a time. Let the competing go by rounds. That is, 5 people compete for 4 chairs in the beginning of each round, and the 4 winners release their chairs at the end of each round so that the next round starts. Please use semaphores to write programs for the entry and exit sections of the 5 people so that every person will win at least one time in competing the chairs for every 3 consecutive rounds.

9 (15%)

If the number of frames allocated to a low-priority process falls below the minimum number required by the computer architecture, we must suspend the execution of that process. We should then page out its remaining pages, freeing all its allocated frames. A process is thrashing if it is spending more time paging than execution.

A. (3 points) For a system supporting virtual memory with unlimited swap space, how do we make a process to be thrashing?
B. (3 points) If we do not allow a thrashing process to steal frames from other processes, will the other processes finish soon and the thrashing process eventually get enough frames to run? Why?
C. (3 points) Why does working set model prevent thrashing?
D. (3 points) Would working set model prevent thrashing totally? Why?
E. (3 points) If processes in a virtual machine are thrashing, how does it affect the performance of the other virtual machines?

見背面

10 (10%)

Vectored I/O or scatter-gather I/O allows one system call to perform multiple I/O operations involving multiple locations.

A. (5 points) Please describe the advantages using vectored I/O with a real application scenario.
B. (5 points) When we implement a system call for vectored I/O, such as `readv` or `writev` in POSIX, what need to be considered to avoid the disadvantages using a vectored I/O.

試題隨卷繳回