Q. 1.6

year	tech	clock speed	ipc/ core	cores	dram bandwidth	sp floating	cahe
2010	32	3.33	4	2	17.1	107	4
2013	22	3.9	6	4	25.6	250	8
2015	14	4.2	8	4	34.1	269	8
2017	14	4.5	8	4	38.4	288	8
2019	14	4.9	8	8	42.7	627	12

10-13	-45%	17%	50%	100%	50%	134%	100%
13-15	-36%	8%	33%	0%	33%	8%	0%
15-17	0%	7%	0%	0%	13%	7%	0%
17-19	0%	9%	0%	100%	11%	118%	50%
Imp/year	-8%	4%	8%	20%	11%	27%	15%
double every	8.80	17.63	8.64	3.60	6.75	2.71	4.80

Q. 1.6

proccessor	clock rate	CPI
p1	3.00E+09	1.5
p2	2.50E+09	1
p3	4.00E+09	2.2

Performance = Clock Rate / CPI		
Number of Cycles = Clock Rate × Time (in seconds)		
Number of Instructions = Number of Cycles × CPI		
Time in sec	10	

	Performance		**number of cycles**	number of instu.
p1 performance	2.00E+09	instruction/second	3.00E+10 cycles	4.50E+10 instructions
p2 performance	2.50E+09	instruction/second	2.50E+10 cycles	2.50E+10 instructions
p3 performance	1.82E+09	instruction/second	4.00E+10 cycles	8.80E+10 instructions

a. p2 has the highest performance

Q 1.7

Column1	CPI A	CPI B	CPI C	CPI D	Clock Rate
P1	1	2	3	3	2.50E+09
P2	2	2	2	2	3.00E+09
	10%	20%	50%	20%	

Instruction count = 1.00E+06

which is faster?

a. what is the global CPI for each

global cpi p1 = 2.6

global cpi p2 = 2

b.Find the clock cylcles for both cases

cpu clock cylcle = Instructions X average clock cycles per instruction(cpi)

clock cyles p1	2.60E+06
clock cyles p2	2.00E+06

CPU execution time = clock cycles / Clock Rate

Execution time P1	1.04E-03 seconds
Execution time P2	6.67E-04 seconds

since P2 has a faster clockrate as well as a lower global Clock cycle per instruction (CPI), it has a shorter execution time.

Q. 1.10.1

Column1	CPI	clock rate	# of instruction	1. clock cycles	2 ins. Time	4. instr. Time	8. inst time
arithmitic	1	2.00E+09	2.56E+09	2.56E+09	1.83E+09	9.14E+08	4.57E+08
Load/Store	12	2.00E+09	1.28E+09	1.54E+10	1.10E+10	5.49E+09	2.74E+09
branch	5	2.00E+09	2.56E+08	1.28E+09	1.28E+09	1.28E+09	1.28E+09
				1.92E+10	**1.41E+10**	**7.68E+09**	**4.48E+09**

		0.7 X # of processor	1.10.1	1.10.2
program is parallel	# of processor	# of instructions can handle	Total execution time for all 3	doubled execution time
	1	0.7	9.60	10.88
	2	1.4	7.04	7.95
	4	2.8	3.84	4.30
	8	5.6	2.24	2.47

by changing the value of b49 the cpi we can see the effect of the execution time reduce to 3.84 do when the load and store is reduced to 3 cpi

Q. 1.14

computer	execution time	fp execution time	l/s execution time	branch time	INT	
original	250	70	85	40	55	
20% better fp	236	56	85	40	55	the total time is reduce to 236 seconds
total reduced 20%	200	70	85	40	5	the int is reduce to 5 second
get to 20% better by reduce branch	200	70	85	-10	55	because we get a negative number it is not possible.

Q 1.15 Clock rate 2.0E+09

Program a	Fp execution	INT	L/S	Branch	total time		
# of Instructions	5.0E+07	1.1E+08	8.0E+07	1.6E+07			
cpi	1	1	4	2			
execution time	0.03	0.06	0.16	0.02	**0.26**		
2x faster fp improvement	-0.10	0.06	0.16	0.02	0.13	-4.12	It would need to be at -4 cpi, but this is not possible.
2x faster l/s improvement	0.025	0.055	0.03	0.016	0.13	0.80	cpi. The cpi of the ls would need to be .8

	0.015	0.033	0.112	0.0112	0.1712	1.495327103 The new time is about 1.5 times faster than the original.

Q. 2.1

c-code	f = g + (h - 5)		temp x0	
Column1	**Column2**	**Column3**	**Column4**	**Column5**
assembly code	addi	x7	x7	-5
	add	x5	x6	x7

Q. 2.2

translate to c code				
assembly	add	f	g	h
	add	f	I	f

c-code	f = I + (g+h)

Q. 2.3

c-code	B[8] = A[I - J]					
f X5	G X6	H X7	I X28	J X29	array a x10	array b x11

assembly code					
	sub	x5	x28	x29	calculae [i-j] and store in x5
	slli	x5	x5	2	multiply j by 4 to get off set
	addi	x6	x5	x10	add offset to array a
	lw	x7	0(x6)		load value of a[i-j] in x28
	addi	x9	x7	32	calculate addre of b[8]
	sw	x7	0(x9)		store value of a[i-j] in b[8]

Q. 2.5

0xabcdef12	show how the value would be arranged In memory little/big indian

big endian

address	0	1	2	3
	ab	cd	ef	12

little endiian

address	3	2	1	0
	ab	cd	ef	12

Q. 2.6 Translate 0xabcdef12 into decimal.

	ab	cd	ef		12				
	10 11	12 13	14 15	1 2					
10	11	12	13	14	15	1	2		
2684354560	184549376	12582912	851968	57344	3840	16	2	2882400018	decimal value

Q. 2.7

B[8] = A[i] + A[j];

f	g	h	i	j	base a	base b
x5	x6	x7	x28	x29	x10	x11

slli 5x, x28, 2	
add x5,x5, x10	//a[i]
lw x7, 0(x5)	
slli x5, x29, 2	
add x5,x5, x10	//a[j]
lw x6, 0(x5)	
add x5, x7, x6	
sw x5, 32(x11)	save b[8] = a[i]+a[j]

Q. 2.10 add x30, x5, x6

registers

x5	0x8000000000000000	
x6	0xD000000000000000	
x30	0x5000000000000000	this is strored in register x30.

Q 2.10.2	This is not the desired value this is because there has been an overflow.		0x15
Q 2.10.3	For sub x30, x5, x6. register x30 contains.		0x5
Q 2.10.4	yes, this is the desired result.		

Q. 2.10.5

add x30, x5, x6	//x30 =	0x5000000000000000	
add x30, x30, x5	//x30 = x30 + x5	x30 results in	0xD800000000000000

Q. 2.13	of the following instruction:	
	sw x5, 32(x30)	
	Hex value =	0x025f2023
	Format	S-type

Q 2.17

Assume the following register contents:
x5 = 0x00000000AAAAAAAA
x6 = 0x1234567812345678

Q 2.17.1

For the register values shown above, what is the value of x7 for the following sequence of instructions?

slli x7, x5, 4	//shift left logical immediate on value in x5 by 4 bits and saving it in x7
or x7, x7, x6	// bit wise or between x7 and x6, stores back into x7.

after slli x7 =	0x0000000AAAAAAAA0	0000101010101010101010101010101010100000000
or with x6	0x1234567812345678	0001001000110100010101100111100001001100010101010110101111000100
result of x7 after or	0x1abefef5b315aff20	0001101010111110111111101111101011001100010101010111111111000100

Q. 2.17.2

slli x7, x6, 4	//shift value by 4 bits
value of x7 after or	0x2345678123456780

Q. 2.17.3

srli x7, x5, 3	//shift right by 3 bits
andi x7, x7, 0xFEF	//and the x7 and oxfef

after srli x7 =	0x00000000155 55555		10101010101010101010101010101
now oxFEF	0x0000000000000 000FEF		000000000000000000111111101111 000000000000000000010101000101
	result of x7 after andi		0x545

Q. 2.21

Assume x5 holds the value 0x00000000001010000. What is the value of x6 after the following instructions?

bge x5, x0, ELSE	//if x5 is greater than or equal to, do else
jal x0, DONE	//when bge, not true execute this line
ELSE: ori x6, x0, 2	
DONE:	

Since bge is greater we jump to ELSE and do the ori x6, x0, 2

x5 =	0x00000000001010000
2=	0x00000000000000010
result of x6=	0x2

Q. 2.22

Suppose the program counter (PC) is set to 0x20000000.

the max jal is	0x3fffffff
the max beq	current pc = -2^15 to pc+ 2^14

Q. 3.1

5ED4 − 07A4	unsiged subtaction

5ED4

- 07A4.

hex value = 5730

Q. 3.6

185–122. Is there overflow, underflow, or neither?

	8 bit decimal	binary
	185	10111001
-	122	01111010
=	63	00111111

Q.3.9

Assume 151 and 214 are signed 8-bit decimal integers stored in two's complement format. Calculate 151 + 214 using saturating arithmetic. The result should be written in decimal. Show your work.

	decimal	invert			
151	10010111	01101000	+1 =	01101001	
214	11010110	00101001	+1 =	00101010	add
				10010011	
			decimal value	147	