EECS 112 & CSE 132, FALL 2016
QUIZ1: Chapters 1 and 2

Student ID:									Name:

Notes:

- calculators / cell phone are not allowed
- Anything outside the boxes will not be graded.
- Empty boxes without an asterisk (*) are to show your work briefly.
- Empty boxes with an asterisk (*) are to show your final result only.
- Students who turn in neat papers in good handwriting tend to get higher grades. The opposite holds as well!

1- Consider a processor at 1 GHz clock rate. The processor executes a program consisting of instructions A, B, and C with following details

Instruction	Count (billion)	CPI
A	1	4
B	2	3
C	3	1

1a) Calculate the total execution time (in seconds) and overall CPI. Leave the CPI as the simplest possible fraction.

(1e9*4+2e9*3+3e9*1)*(1e-9)=1*4+2*3+3*1	* 13 sec
(1/6)*4+(2/6)*3+(3/6)*1	* 13/6

1b) What should be the new value of CPI of instruction A to get an overall execution time of 10 seconds? What is the new overall CPI in this case?

10=1*x+2*3+3*1=9+x	* x=1
(Total number of clocks)/(total number of instructions)	* 10/6

1c) Parallelism: In 1a, if instructions A, B, and C are independent, and the program is rewritten such instructions A, B, and C are executed on cores 1, 2, and 3 respectively, what is the new total execution time and CPI?

Time on core 1 = 1*4 = 4 seconds Time on core 2 = 2*3 = 6 seconds Time on core 3 = 3*1 = 3 seconds Total time = bottleneck = 6 seconds	* 6 seconds
CPI = (tot # of cycles)/(tot # of inst)=6/6	* 1

1d) Parallelism: In 1a, if instructions A and C are executed on core 1, and instruction B is ran on core 2 independently, what is the new total execution time and CPI? Leave the CPI as the simplest possible fraction.

Total time on core 1 = 1*4+3*1 =7 seconds Total time on core 2 = 2*3 = 6 seconds total time = bottleneck = 7 seconds	* 7 seconds
CPI = (tot # of cycles)/(tot # of inst)=7/6	* 7/6

2- Translate the following C code to LEGv8 assembly. Assume that the values of i, a, and j are stored in X0, X1, and X2. Also, the base address of arrays A, B, and D are stored in X3, X4, and X5. **Do not use MUL**. Comments are mandatory.

```
for(i=0 ; i<a; i++)
    for(j=0;j<i;j++)
        D[3*j]=A[i]+B[j+1]
```

Labels:	Instruction	Comment
	ADDI X0,XZR,#0	Set i=0
LOOP1	SUBIS XZR,X0,X1	Calculate i-a and set flags
	B.GE EXIT1	If i>=a, exit loop
	ADDI X2,XZR,#0	If i<a, set j=0
LOOP2	SUBS XZR,X2,X0	Calculate j-i and set flags
	B.GE EXIT2	If j>=i, exit loop
	LSL X6,X0,#3	X6=8i, the bye offset
	ADD X6,X6,X3	X6=&A[i]
	LDUR X6,[X6,#0]	X6=A[i]
	LSL X7,X2,#3	X7=8j, the byte offset
	ADD X7,X7,X4	X7=&B[j]
	LDUR X7,[X7,#8]	X7=B[j+1]
	ADD X7,X7,X6	X7=A[i]+B[j+1]
	ADD X8,XZR,X2	X8=j
	ADD X8,X8,X2	X8=2j
	ADD X8,X8,X2	X8=3j
	LSL X8,X8,#3	X8=8*3j, the byte offset
	ADD X8,X8,X5	X8=&D[3j]
	STUR X7,[X8,#0]	D[3j]=A[i]+B[j+1]
	ADDI X2,X2,#1	j++
	B LOOP2	Back to inner loop
EXIT2	ADDI X0,X0,#1	i++
	B LOOP1	Back to outer loop
EXIT1		