Tutorial 5
CSE 112 Computer Organization

Note to TAs (Q1) : Solve this first question to explain to students how to write assembly code with focus on how to deal with branches using labels. Explain briefly how to write assembly code for branching and loops using labels.

Q1 a.

```
r0 = 10;
r1 = 5;
r3 = r0-r1
if(!r3)
{
    r2 = 1;
}
else
{
    r2 = 0;
}


-----------------------------
mov r0, #10
mov r1,  #5
sub r3, r1,
r0 bz     r3,
equal mov r2,
#0 bnz r3,
exit
equal :    -----> label
     name mov r2 , #1
exit:      ------> label name
```

Q1 b

```
r0 = 10
r1 = 5
r2 = 1
r3 = 0
If (!r0 ||
     !r1): r3 =
     0
     Jump to
end while(r1):
     r3 +=r0
     r1 -=
     r2
end
 -----------------------------
mov r0, #10
mov r1, #5
mov r2, #1
mov r3, #0
bz r0, arg_zero
bz r1, arg_zero
bnz r2,
arg_zero_end
arg_zero:
     mov r3, #0
     bnz r2,
     end
arg_zero_end
: loop:
     bz  r1,
     loop_exit add
     r3, r3, r0 sub
     r1, r1, r2 bnz
         r2, loop
loop_exit
: end:
```

Q1 c:

```
r0 = 2;
r1 = 0;
r2 = 1;
r3 = 1;
while (r0)
{
     r1=  r1+r2;
     r2=  r2+r3;
     r0= r0-r3;
}

 -----------------------------
 mov r0, #2
 mov r1, #0
 mov r2, #1
 mov r3, #1
 bz  r0,loop_exit
 loop:
     add  r1,  r2,
     r1 add r2, r3
     ,r2  sub  r0,
     r0,   r3  bnz
     r0, loop
loop_exit:
```

Q2.

A.

```
0 mov r0, #2
1 mov r1, #0
2 mov r2, #1
3 mov r3, #1
4 bz  r0,
  loop_exit loop:
5     add r1, r2, r1
6     add r2, r3 ,r2
7     sub r0, r0, r3
8     bnz r0,
 loop loop_exit:
```

Cycle	PC	r0	r1	r2	r3
0	0	2	-	-	-
1	1	2	0	-	-
2	2	2	0	1	-
3	3	2	0	1	1
4	4	2	0	1	1
5	5	2	1	1	1
6	6	2	1	2	1
7	7	1	1	2	1
8	8	1	1	2	1
9	5	1	3	2	1
10	6	1	3	3	1
11	7	0	3	3	1
12	8	0	3	3	1
13	END	END	END	END	END

B.

13 * 1ns = 13ns

C.

13+2+1+1+1+1+2+1+1+1+1+2 = 25cycles = 25*0.5ns = 12.5ns.

Loss in execution time.

NOTE to TAs : The objective of this question is to familiarize students with how code is executed on simple machines. Explain briefly the steps involved in the execution cycle, execution time. You might need to explain why different instructions can take the different number of cycles to execute.

Q3.

A. No it wouldn't. The hardware/ISA decided the opcode format. The specific "flavour" of the syntax is not determined by the hardware/ISA. We can have multiple assembly syntaxes for the same ISA. Example: x86 ISA has 2 widely used syntaxes: the Intel syntax and AT&T syntax.
B. Yes, the assembler needs to be modified so that it can correctly parse the new syntax and generate the correct code.
C. No we can't. Regardless of the syntax, the assembled code will be in the form of binary instructions, whose syntax is dictated by the ISA. No matter what syntax we use, we will get the same binary representation of the program.

   Note to the TAs: The objective of this question is to make students realize that the ISA and the assembly syntax are two different things.