302558184  
302650072  
203090051  
029998937

**Appendix C**

Answer the following questions.

C.1) What are the limitations due to the pipeline latency of the following combinations (assume Data Forwarding already exists):

* beq after add where the add Rd is the beq Rt
* beq after lw where the lw Rt is the beq Rs

Use a similar figure to Fig.2 and Fig. 3 to demonstrate your answers.

WB

EX

ID

IF

WB

EX

ID

IF

WB

EX

ID

IF

NOP

NOP

add **$3**,$5,$8

WB

EX

ID

IF

Beq $9, $3, label

WB

EX

ID

IF

CK

MEM

MEM

MEM

MEM

MEM

EX

WB

EX

ID

IF

WB

EX

ID

IF

WB

EX

ID

IF

NOP

NOP

lw **$3**,16($10)

WB

EX

ID

IF

Beq $9, $3, label

WB

EX

ID

IF

CK

MEM

MEM

MEM

MEM

MEM

EX

C.2) What are the limitations of all cases of C.1 after you add the Branch Forwarding?

For beq after add:

WB

EX

ID

IF

WB

EX

ID

IF

WB

EX

ID

IF

NOP

Beq $9, $3, label

add **$3**,$5,$8

WB

EX

ID

IF

WB

EX

ID

IF

CK

MEM

MEM

MEM

MEM

MEM

EX

For beq after lw: The same as C1.

C.3) Why can’t we check the result of the previous instruction (time slot n-1) by a beq instruction following it (time slot n)?

Because the (n-1) instruction EX phase and the beq ID phase are executing concurrently, and beq might not get the correct value from the EX phase. We need a NOP, for letting the EX phase complete and then forward the value.

C.4) What is the shortest loop code possible? Any limitations? Explain in detail

LABEL: Add $2, $3, $4

nop

Beq $9, $10, LABEL

Nop

The limitations are:

1. We need to execute a calculation, which is labeled, for the loop
2. Before beq we need a NOP, because branch forwarding makes forward the data from the MEM phase of add, to ID phase of beq.
3. Beq itself
4. Need a NOP for not executing the next instruction twice.