1. Here is the space-time diagram execution of given Assembly program.



Our program executes loop twice for R3 = 0x02. Stalls are indicated as “—” (double dash). I will explain conflicts below:

1. XOR R3, R3, R3,  
   ADD R3, #$1000, R1

There is a data dependency so we need to stall pipeline only for 1 cycle thanks to ME/WB operand forwarding.

1. LDL $0(R1), R4  
   LDL $0(R2), R5  
   XOR, R4, R5, R4

Here there is a load-use hazard. We don’t need to stall for R4 since it gets ready while R5 executes but we need to stall for R5. So 1 cycle here.

1. XOR, R4, R5, R4  
   XOR R4, R5, R5

XOR R4, R5, R4

STL $0(R1), R4

We have triple dependency here so we are stalling for 3 cycles.

1. SUB R3, #1, R3

BNZ LOOP

~~BRU DONE~~

~~ADD R1, R2, R1~~

ADD R1, #4, R1

LDL $0(R1), R4

We have a control hazard here. 2 instructions come after BNZ are flushed and we fetched LDL instead of gray instruction when BNZ took branch. So we have 2 cycle stall here.

We have same stalls stated in 2 and 3 for loop.

1. BNZ LOOP

BRU DONE

~~ADD R1, R2, R1~~

~~ADD R1, #4, R1~~

DONE SUB R3, R3, R3

We have 2 cycle stalls again for taking a branch. After exiting loop, we took unconditional branch DONE.

So for total we have:

We loop 2 times here so total stall is:

1. I did some pre work in part A so I will be using that.

Before entering loop, we have 1 cycle stall. Inside loop we have 4 cycle stall XOR operations. For branching to next iteration, we have 2 cycle stalls. For exiting iteration, we also have 2 cycle stalls. For each n iterations we have 6n stall. So formula is:

1. CPI for given program: