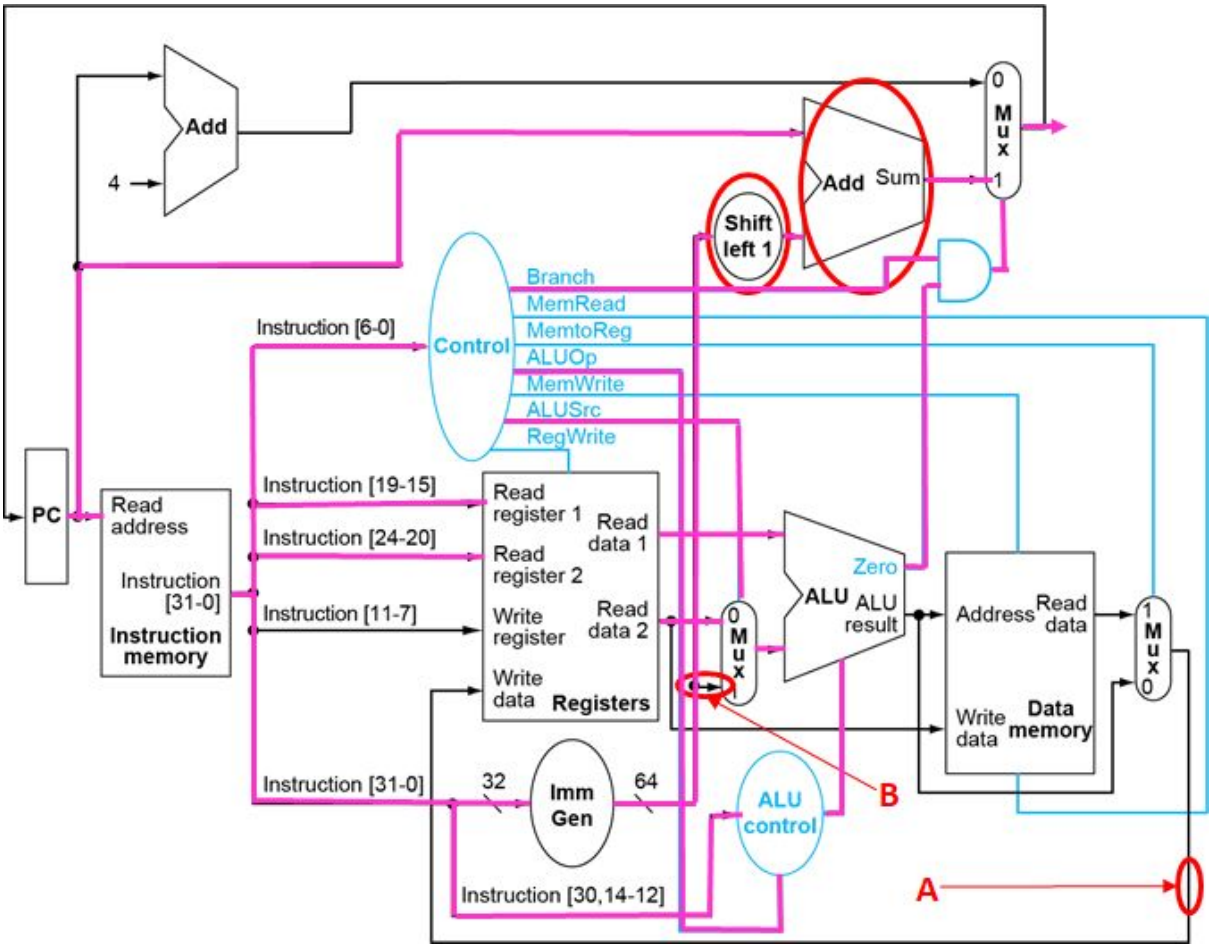
1. beq x3, x5, -496
   (a) -496
   (b) 9900 8700 0080 9CE0
   (c) Control (由上至下): 1, 0, 0, 01 (sub), 0, 0, 0
   ALU control: 0110 (sub)
   Zero: 1
   (d) 9900 8700 0080 9CE0



   (e) 不變
   (f) 為了要乘以2，因為branch及jump指令的immediate不含最後一個bit
   (g) add, sub, slt, ld
   (h) sd, ld
   (i) 只要不是從memory讀到register的指令都會掛掉，例如add
   (j) 針對beq這個instruction，Instruction memory, Registers, ALU act as combinational circuits，因為它們在該cycle只會根據input去給出output，並沒有寫入的情形發生。

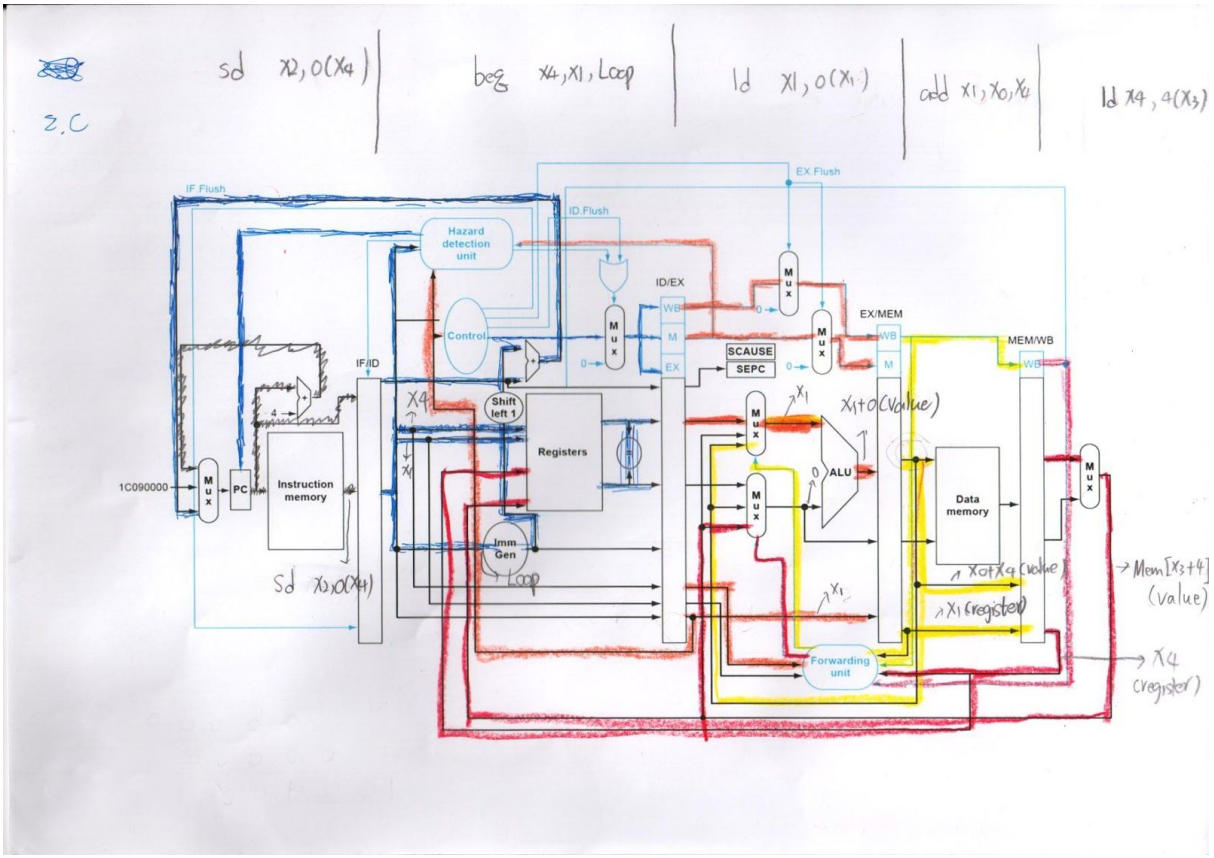
2. (a) ld x4,4(x3)
   NOP
   NOP
   add x1,x0,x4
   NOP

NOP
ld x1,0(x1)
NOP
NOP
beq x4,x1,Loop
(NOP
NOP
NOP)
[註: 此處的branch可能taken or not taken，而為了讓CPU在branch taken的情況下執行正確，理論上要加上NOPs，但是因為助教在之前的討論區說明可以只考慮data hazards，所以寫3個NOPs或不寫都給對。]
sd x2,0(x4)

(b)
ld x4,4(x3)
NOP
add x1,x0,x4
ld x1,0(x1)
NOP
NOP

beq x4,x1,Loop
(NOP)
[註: 在有hazard detection但是沒有stall的情況之下，CPU可以處理branch taken之後的control hazard，所以可加可不加]
sd x2,0(x4)

(c) 要在path上標出每個instruction的值，少一個扣一分



(d) 以下錯誤至少寫出一種，若寫的不完整扣一半分數:

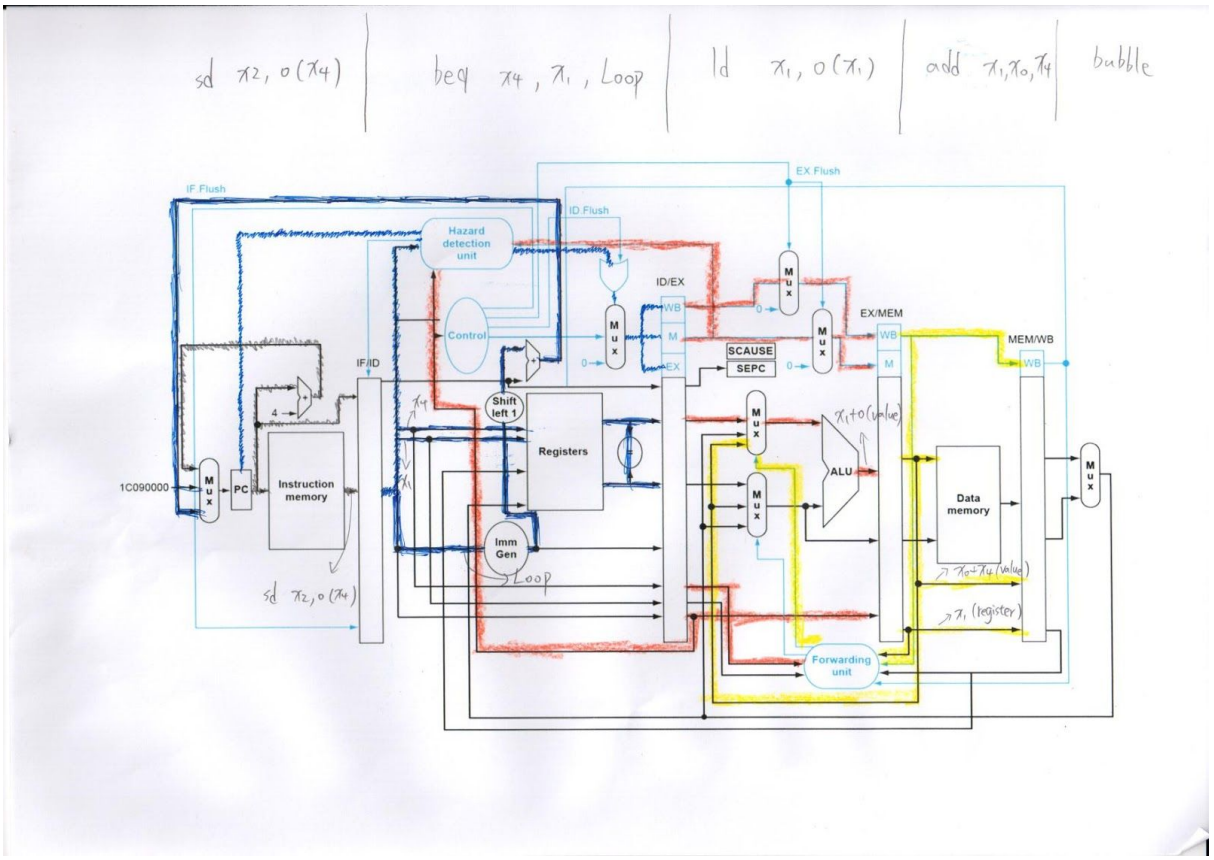
In MEM stage, add x1,x0,x4 所算出來的x1的值是錯的。

In EX stage, the forwarding path from Mem to Ex for x1 is wrong, 所以ld所使用的x1的值是錯的。

In ID stage, x1的值還沒被更新，因此beq的結果可能是錯的。

In IF stage, 可能會有controll hazard, load錯誤的指令

(e)



3. IF: 400ps
   ID: 230ps
   EX: 160ps
   MEM: 350ps
   WB: 240ps
   minimum cycle time of the computer => 400ps