Het gebruikte programma waarin zich een hazard voordoet is een versimpelde versie van het meegeleverde bestand: hazardtest.wasm. (Toestemming van tutor)

1 a

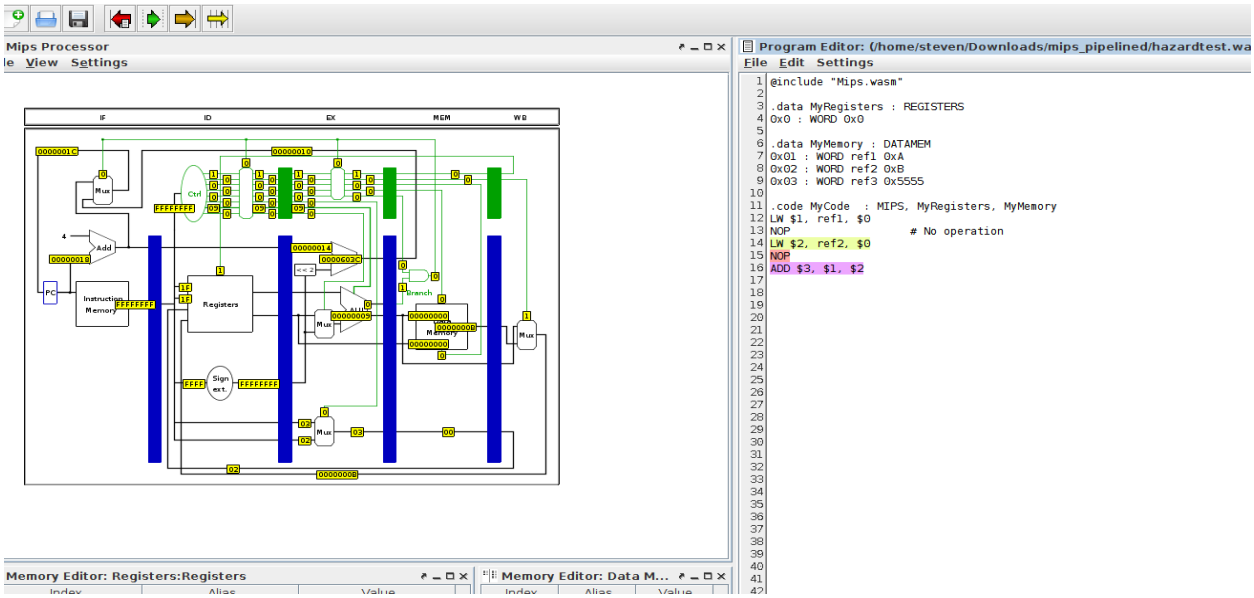
In dit programma wordt eerst referentie1 in register $1 geplaatst (via LW). Vervolgens wordt referentie 2 in register $2 geplaatst en tot slot wordt de som hiervan opgeslagen in register $3. Omdat register $2 nog niet de opgegeven LW-waarde heeft (door toedoen van de pipeline) wordt er gerekend met -1 (onafhankelijk van programma). De som van register $1 en $2 zal dus altijd $1-1 geven in dit geval.

1 b

Bij dit programma is er een stal toegevoegd (NOP-instructie) tussen de eerste LW- en de tweede LW instructie zodat het programma register $2 de juiste waarde kan doorgeven aan register $2 voordat hiermee gerekent kan worden door de ALU bij de add-instructie, zodat de add-instructie uiteindelijk ook het gewenste resultaat geeft.

1 c

We hebben hier te maken met een Data Hazard. De add instructie heeft namelijk eerst de waardes van $1 en $2 nodig voordat hij deze moet optellen. In het geval waar de data-hazard niet is opgelost, telt hij de waardes $1 en $2 al bij elkaar op voordat $2 de juiste waarde verkregen heeft.



2 a

De waarde van register $2 wordt zodra deze verkregen wordt direct doorgestuurd naar de ADD-instructie maar voert eerst nog de eerst volgende instructie uit. Wanneer de ADD-instructie dus direct voor de LW instructie van register $2 zou staan zou het nog steeds misgaan. Echter staat er nog een NOP instructie tussen. De architectuur voert dus eerst nog de NOP instructie uit terwijl hij de waarde van register $2 alvast doorstuurt naar de ADD-instructie

2 b

In dit geval geeft de forwaring-architectuur een beter resultaat zonder de forwarding. Dit omdat de forwarding architectuur een enkele instructie minder nodig heeft (één NOP) om tot hetzelfde resultaat te komen als de architectuur zonder forwarding. Dit is ook terug te zien in het CPI van de computer.