**Q1.5** -> Os valores no registo x11 mantém-se inalterado, enquanto o valor de x13 é alterado de 0x10000030h para 0x0000030h. Isto deve-se, pois, a instrução que definirá o valor de x13 (addi x13, x11, 48) contém um registo que ainda não foi escrito na memória (registo x11) e isso leva a um conflito de dados. Assim, para a execução desta instrução o valor usado para x11 é o valor que estava na memória naquele momento (valor com que x11 é iniciado no programa que é 0x0000000h).

**Q1.6**

|  |  |  |
| --- | --- | --- |
| **Instrução que escreve** | **Instrução que lê** | **Operando que provoca conflito** |
| addi x11, x3, 0 | addi x13, x11, 48 | X11 |
| addi x13, x11, 48 | addi x12, x13, -4 | X13 |
| add x20, x13, x16 | lw x21, 0(x20) | X20 |
| lw x22, 0(x11) | add x22, x22, x23 | X22 |
| lw x23, 0(x12) | add x22, x22, x23 | X23 |
| add x22, x22, x23 | mul x15, x15, x22 | X22 |
| sub x22, x12, x11 | srai x22, x22, 2 | X22 |
| sub x22, x12, x11 | add x14, x14, x22 | X22 |
| srai x22, x22, 2 | add x14, x14, x22 | X22 |
| addi a7, x0, 10 | ecall | X17 |

**Q1.8** -> Depois da resolução dos conflitos de dados antes do ciclo “while” o valor dos dois registos mantém-se iguais nas duas arquiteturas. Isto já era expectável visto que se não há conflitos, então a mudança entre os dois não deve alterar os valores.

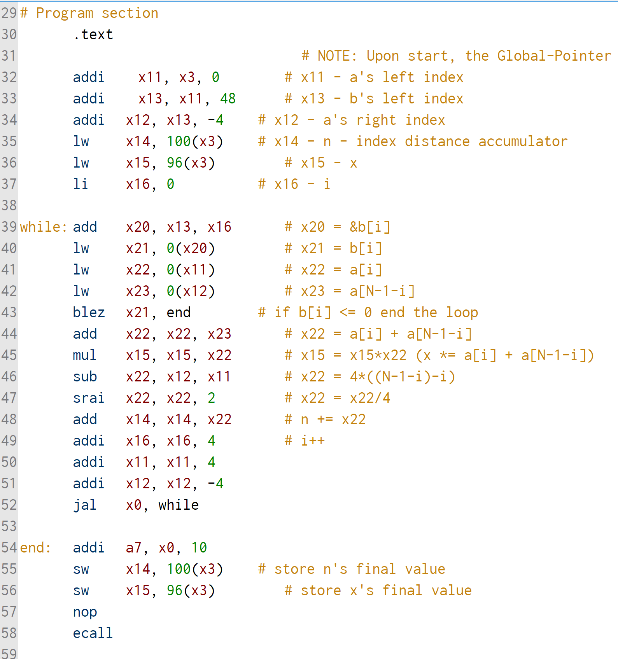
**Q1.11**

Número de ciclos de relógio: 145

Número de instruções executadas: 125

**Q1.12** ->

**Q1.13** -> A política de execução de salto é a política de predict not taken, ou seja, o ripes assume sempre que no IF a instrução não é um branch. Assim, quando este chegar ao exe (onde está o controlador de branch) caso haja um branch ele descarta as duas instruções novas que entraram (dá flush) e salta para o pc indicado pelo branch.



**Q2.3** -> As alterações feitas foram:

* Reorganização do código nas linhas 41, 42 e 54, pois evita ter de se colocar mais nop. Isto deve-se, pois, o x21 tem de ter o valor guardado para fazer o branch e este só é guardado na fase da mem e, portanto, só quando x21 está no wb é que o branch pode ser executado.
* Adição de um nop na linha 57 para pois para o ecall o valor de x17 tem de estar a 10, logo precisa de três instruções pelo meio.

**Q2.4**

Número de ciclos de relógio: 134

Número de instruções executadas: 114

**Q2.5 ->** . Comparando este valor com o da pergunta 1.12 podemos concluir que esta arquitetura é mais eficiente.

**Q2.6 ->** 1.08

**Q3.3 ->** Nenhuma, pois todos os problemas foram resolvidos por hardware.

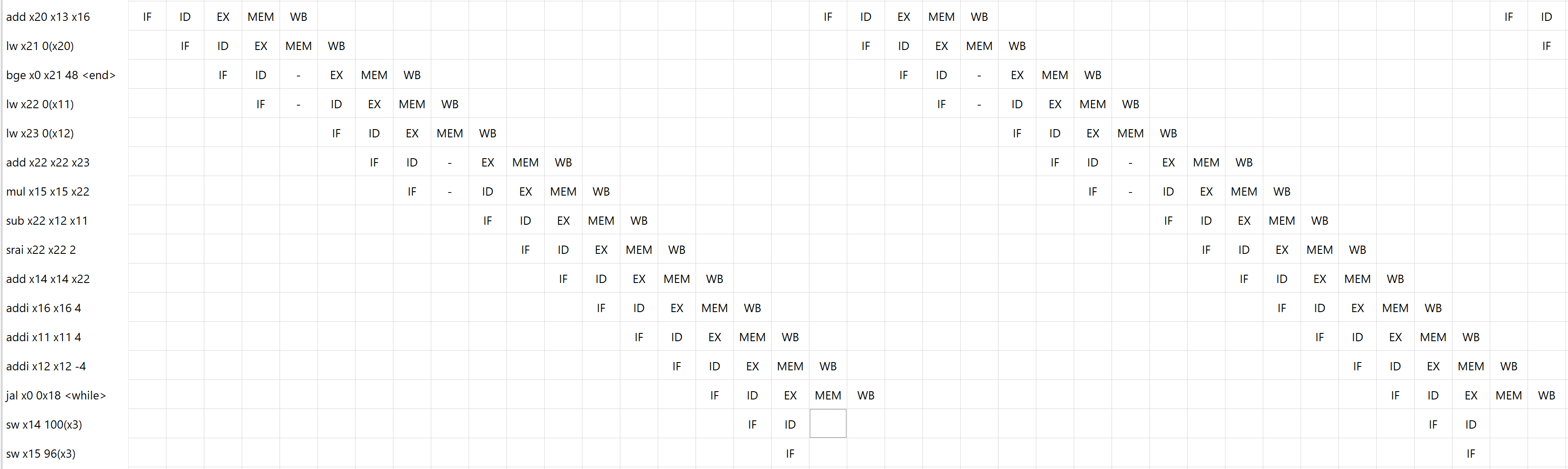
**Q3.4**

Número de ciclos de relógio: 148

Número de instruções executadas: 111

**Q3.5 ->** . Comparando este valor com o da pergunta 1.12 e o da pergunta 2.5 podemos concluir que esta arquitetura é mais eficiente.

**Q3.6 ->** ciclo. Este valor é menor que 1 pois existem stall pelo meio o que faz com que as instruções demorem mais que um ciclo a ser concluídas. Para além disso esse número também diminui com os flush dos branch e dos jal.

**Q3.9**

**Q3.10**

Stall conflitos de dados raw = 4

Stall conflito de controlo = 0

**Q3.11** O IPC é inferior a 1 porque há instruções em que acontece um flush devido aos saltos dos branch e dos jal e porque existe um stalls entre o lw x21 0(x20) e o branch. Este stall deve-se ao facto de o load para x21 só ir buscar o valor à memória na parte da MEM e depois só pode ser usado em WB. Logo o EX do bge tem de ser dado quando o load já está no WB.

**Q4.2 ->** As alterações feitas foram:

Reorganização do código nas linhas 42, 43 e 59, pois assim evita-se de usar stall para esperar o x21 para o branch e o x17 para o ecall.

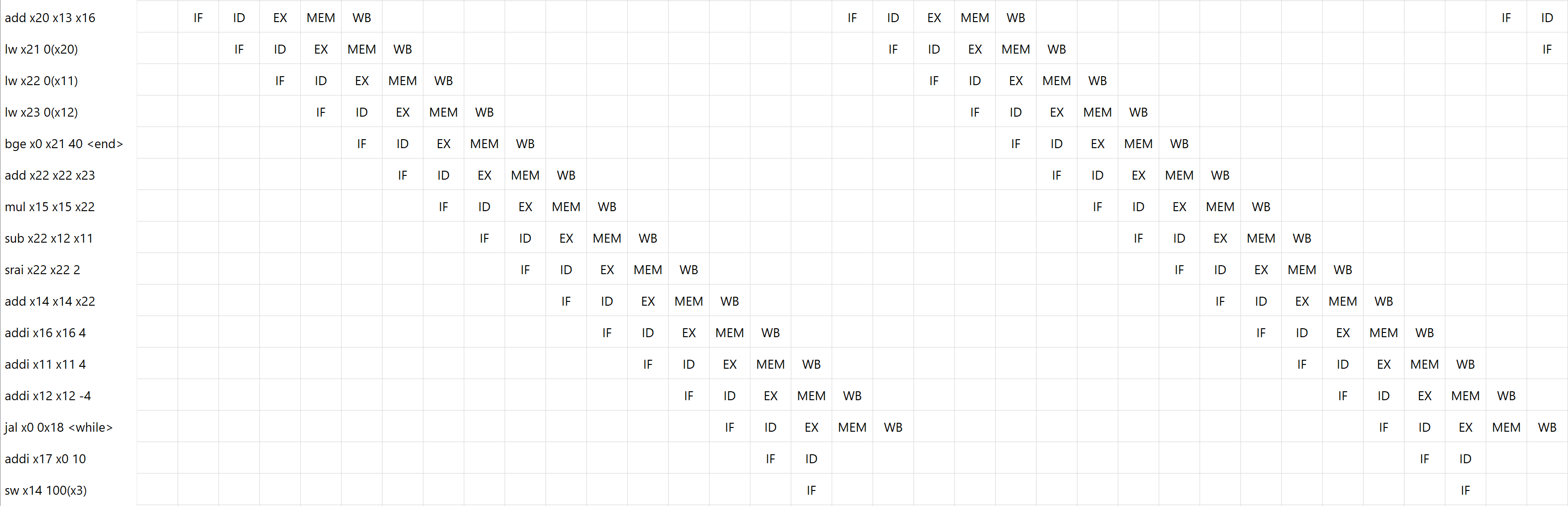
**Q4.3**

Número de ciclos de relógio: 133

Número de instruções executadas: 113

**Q4.4->** ciclo. Comparando este valor com o anterior podemos ver que este sobe em 0.1, devido a já não ser necessário o uso de alguns stalls.

**Q4.7**



**Q4.8**

*Stall* conflitos de *raw* = 0

*Stall* conflito de controlo = 0

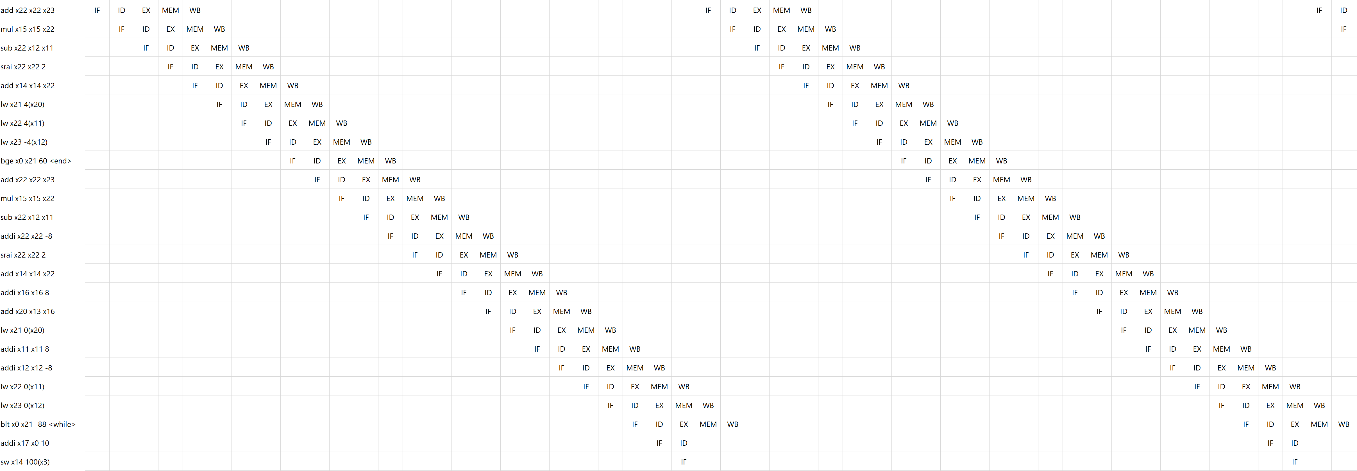
**Q4.9 ->** O IPC continua inferior a 1 porque há instruções em que acontece um flush devido aos saltos dos *branch* e dos *jal*.

**Q6.2**

Número de ciclos de relógio: 105

Número de instruções executadas: 93

**Q6.3 ->**. Este parâmetro aumenta em 0.036, pois ao fazermos um loop unrolling diminuímos a quantidade de instruções de salto no programa, logo perdemos menos instruções para *flush* ao longo do programa.

**Q6.6**

**Q6.7**

*Stall* conflitos de *raw* = 0

*Stall* conflito de controlo = 0

**Q6.8 ->** O IPC continua inferior a 1 porque há instruções em que acontece um flush devido aos saltos dos *branch*.

**Q6.9->** 1.27