Laboratório 4 de

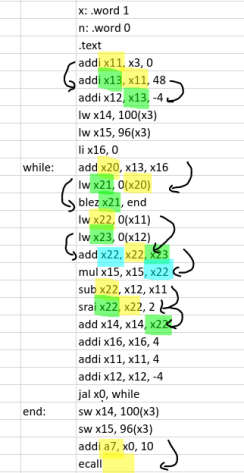
Arquitetura de Computadores

Exercício 1

Q1.5.

Os valores não são iguais, porque há conflitos de dados.

Q1.6.

Registos com cores iguais em instruções consecutivas indicam os registos que causam os conflitos.

Q1.7.

Ficheiro lab4\_01.s

Q1.8.

Agora, os valores são 0x10000000 para o x11 e 0x10000030 para o x13, o que corresponde aos valores de início dos vetores.

Q1.9.

Ficheiro lab4\_02.s

Q1.11.

12x7 + 9x8 + 3 + 9 = 168 instruções

188 ciclos de relógio

Q1.12.

3x8 + 3x7 + 3 = 48 instruções inúteis

168 - 48 = 120 instruções úteis

rácio: 48/120 = 0,286

considerando T original = 5xT pipeline

111x5 = 555

S = 555/168 = 3.304

Logo, ao utilizarmos esta nova execução tornamos o programa mais eficiente com um speedup, no caso ideal, de 3.304.

Q1.13.

A política de predição utilizada é um preditor de salto estático do tipo NOT TAKEN, isto porque, ao ser verificada a condição onde vai haver um salto, as 2 instruções que já se encontram em pipeline são eliminadas.

Exercício 2

Q2.3.

Foi colocada a instrução "addi a7, x0, 10" no início do programa (que estava antes do ecall), para não haver conflitos de dados com o ecall.

As instruções “lw x22, 0(x11)” e “lw x23, 0(x12)” foram colocadas antes do “blez” para evitar conflitos de dados, isto porque, como descrito no enunciado, as instruções de controlo, não usufruem do forwarding, o que leva á necessidade da utilização de stalls, desta forma, ao colocarmos as instruções nesta posição substituímos os stalls.

Ficheiro lab4\_03.s

Q2.4.

5x8 + 9x7 + 7 + 3 = 113 instruções

133 ciclos de relógio

Q2.5.

Rácio: 113/113 = 1, não foram utilizados nops

considerando T original = 5xT pipeline

111x5 = 555

S = 555/113 = 4.912

Logo, ao utilizarmos esta nova execução tornamos o programa mais eficiente com um speedup, no caso ideal, de 4.912.

Q2.6.

S = 168/113 = 1.487

Logo, ao utilizarmos a última arquitetura tornamos o programa mais eficiente com um speedup, no caso ideal, de 1.487.

Exercício 3

Q3.3 Não foram introduzidas alterações pois esta arquitetura resolve todos os conflitos, apesar de uma forma menos eficiente

Q3.4.

111 instruções

148 ciclos de relógio

Q3.5.

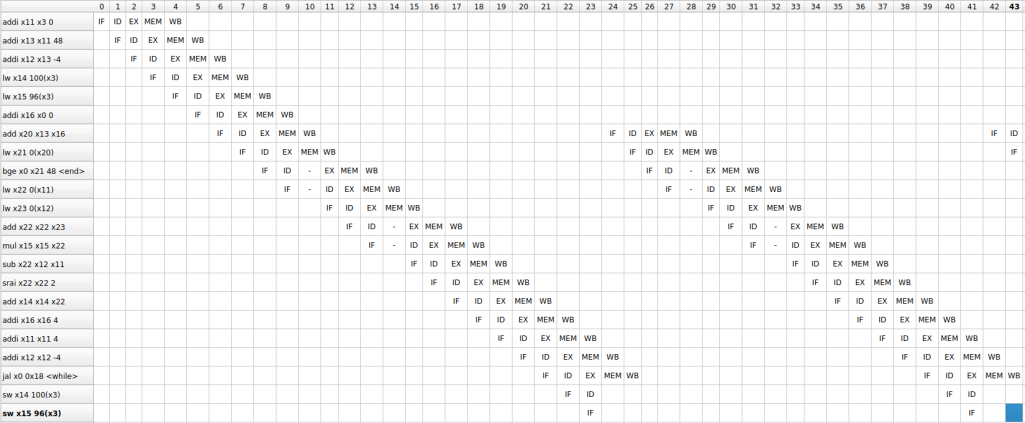
Como não foram inseridos nops, não há instruções não uteis, apenas stalls.

Logo, o rácio será 1.

Q3.6.

IPC = 0.75, este valor é menor que 1 porque cada instrução corresponde a 5 ciclos, mas, por ser uma arquitetura em pipeline, por cada ciclo de relógio, uma instrução é executada, no entanto, no fim do programa não haverá mais nenhuma instrução a executar, para além de outros casos onde são executados mais ciclos de relógio (flush e stall).

Q3.9.



Q3.10.

Há 8 stalls introduzidos em consequência de conflitos de dados.

Não são implementados stalls para os conflitos de controlo.

Q3.11.

A existência de stalls aumenta o número de ciclos realizados, que é inversamente proporcional ao IPC, logo o IPC diminui, sendo assim menor que 1.

Exercício 4

Q4.2.

Foram colocados 2 “lw” antes do “blez”, isto porque, os stalls que ocorriam eram devidos a conflitos de dados relacionados com a escrita em memoria nos “lw”.

Ficheiro lab4\_04.s

Q4.3.

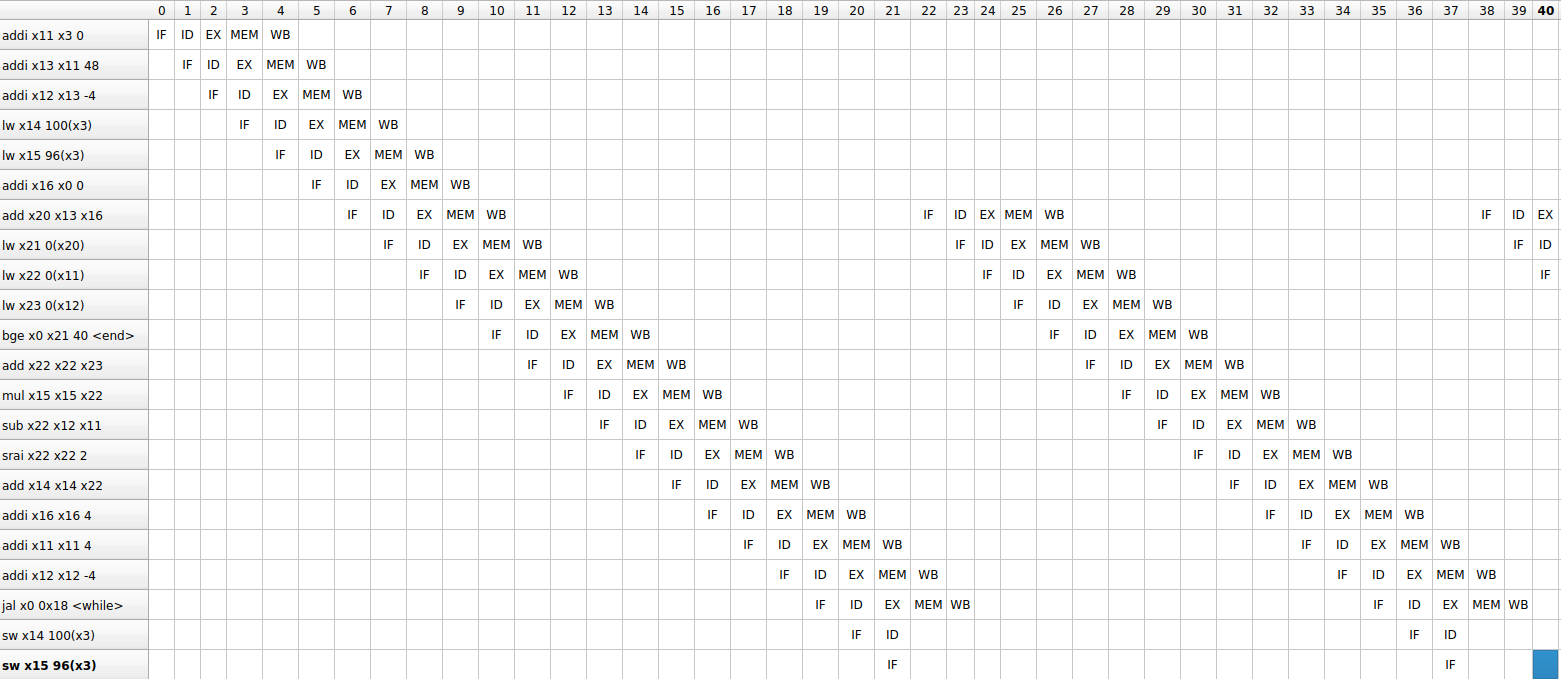
135 ciclos de relógio

113 instruções

Q4.4.

IPC = 0.837, este valor aumentou, pois, ao ser reorganizado o código para não haver existência de stalls, aumentámos o número de iterações em 2 instruções, para além da diminuição do número de ciclos nas situações onde ocorreriam stalls.

Q4.7.



Q4.8.

Não existem stalls, na reorganização fizemos com que não houvesse mais stalls.

Q4.9.

Estando em arquitetura de pipeline a cada ciclo, é iniciada uma nova instrução e uma instrução é acabada de ser executada. No entanto, no fim do programa e em algumas situações em específico, flushes e stalls, mias ciclos irão ocorrer. No final do programa 4 ciclos serão realizados para a última instrução acabar de ser executada. No caso dos flushes e stalls há mais ciclos que serão realizados para as operações ocorrerem, no caso deste programa os stalls não ocorreram, pois foram retirados com a reorganização do código, mas os flushes ocorrem nos conflitos de controlo.

Exercício 5

Q5.3.

Ficheiro lab4\_05.s

O objetivo das alterações foi fundir o maior número de instruções possível e também colocar o maior número de instruções possíveis a seguir às instruções de controlo, diminuindo assim o número de ciclos.

Q5.4.

109 ciclos de relógio

84 instruções

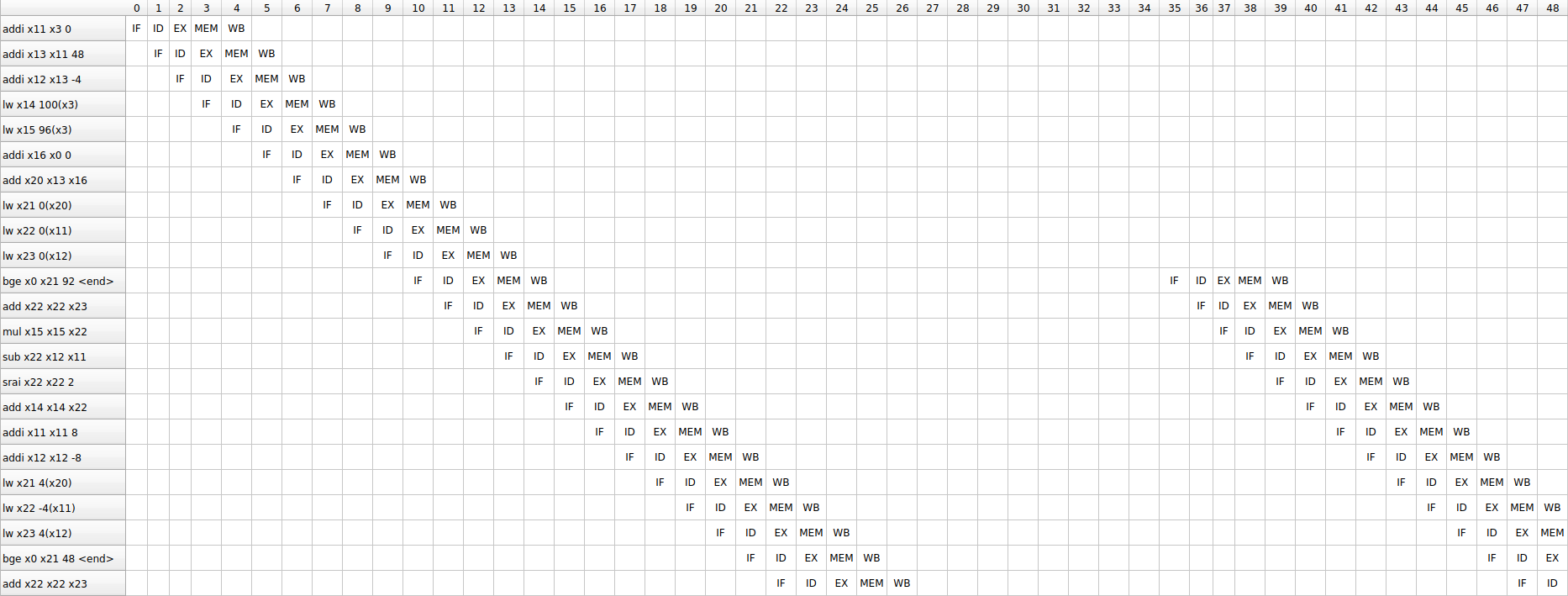
Q5.5.

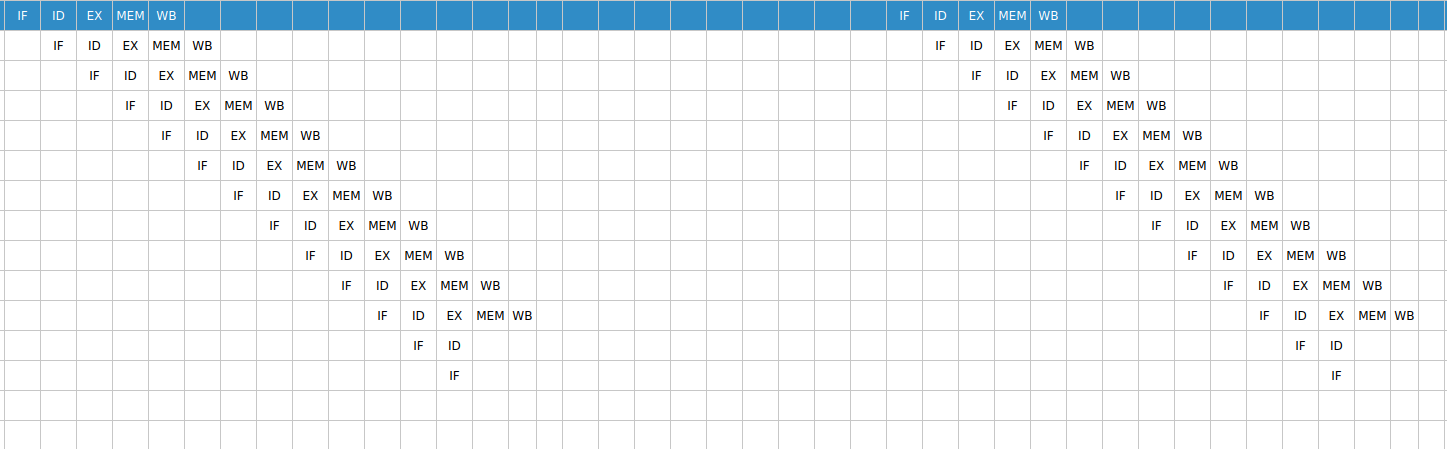
Rácio = 1. Não foram utilizados nops.

Q5.6.

IPC = 0,872, este valor é menor que 1 porque cada instrução corresponde a 5 ciclos, mas, por ser uma arquitetura em pipeline, por cada ciclo de relógio, uma instrução é executada, no entanto, no fim do programa não haverá mais nenhuma instrução a executar, para além de outros casos onde são executados mais ciclos de relógio (flush).

Q5.9.





Q5.10.

Não há stalls introduzidos para conflitos de dados e também não são implementados stalls para resolver conflitos de controlo.

Q5.11.

Devido a conflitos de controlo há um maior número de ciclos a ser executados que não correspondem a instruções completas, para alem de que no fim do programa os últimos 4 ciclos correspondem à mesma instrução.