**Q1.3**

**&a[0] = 0x10000000**

**&b[0] = 0x10000030**

**Q1.5**

**Não porque quando a 2ª instrução acede ao banco de registos, o registo x11 ainda não esta atualizado porque a 1ª instrução ainda não chegou ao WB (write back) (CONFLITO DE DADOS).**

**Q1.6**

**Table

Description automatically generatedTable

Description automatically generated**

**Q1.8**

**Os valores são iguais aos que tínhamos previsto na pergunta 3, e são diferentes dos valores que estavam antes de ser corrigido os conflitos de dados antes do “while”. Como o x11 tem o valor de x3, para evitar ficar mais ciclos de relógio a espera, a 2ª instrução pode ir buscar o valor de x3 em vez do valor de x11. Para corrigir o 2º conflito de dados basta mudar a 3ª instrução para depois dos “lw” para que o ID desta seja após o WB da 2ª instrução. Como o x16 é necessário na 1ª instrução do “while”, mete se este antes dos “lw”.**

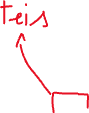
**Q1.11**

**Clock cycles = 152**

**Instruction = 132**

**Graphical user interface, application, table

Description automatically generatedQ1.12**



**A execução do programa e bastante eficiente pois só tem 15% de instruções não uteis**

**Q1.13**

**Predict not taken (usa um invalidate no EX)**

**A picture containing text

Description automatically generatedA picture containing text, document, receipt, screenshot

Description automatically generatedQ2.3**

**ANTES DEPOIS**

**Q2.4**

**Clock cycles = 136**

**Instruction = 116**

**Graphical user interface, application, table

Description automatically generatedQ2.5**

**Rácio instruções NÃO UTEIS = 0,03**

**Rácio instruções UTEIS = 0,97**

**Apenas não deu para tirar os nops antes do ecall, por isso consideramos o nosso programa bastante eficiente**

**Q2.6**

**Speedup = 152 / 136 = 1,11**

**Q3.3**

**NONE porque o forwarding e o hazard corrigem os conflitos de dados**

**Q3.4**

**Clock cycles = 148**

**Instruction = 111**

**Q3.5**

**Rácio Instruções uteis = 1**

**Rácio Instruções não uteis = 0**

**Muito eficiente**

**Q3.6**

**IPC = 0,75**

**E menor que 1 porque há instruções com mais de 1 ciclo de relógio.**

**Q3.9**

**Chart, scatter chart

Description automatically generated**

**Q3.10**

**Stalls conflitos dados RAW = 4**

**Stalls conflitos controlo = 0**

**Q3.11**

**Stalls => mais ciclos por instrução**

**Q4.2**

**Mudar os ‘lw’ para antes do branch**

**Q4.3**

**Clock cycles = 135**

**Instruction = 113**

**Q4.4**

**IPC = 0,837**

**O valor do IPC e mais alto, logo o processador e mais rápido pois executa o mesmo numero de instruções com menos ciclos de relógio, mantendo a mesma frequência do clock.**

**Q4.7**

**Chart, scatter chart

Description automatically generated**

**Q4.8**

**Stalls conflitos dados RAW = 0**

**Stalls conflitos controlo = 0**

**Q4.9**

**Foi possível anular completamente os stalls.**

**Q5.3**

**A picture containing text

Description automatically generated**

A drawing of a building

Description automatically generated with medium confidence

**Os lw da 2ª iteração passam a ter um offset para não termos de usar adds antes.**

**Na 2ª iteração o ‘n’ e incrementado com o valor da iteração anterior menos 2.**

**Os addi no final passam a 8 em vez de 4.**

**Q5.4**

**Clock cycles = 110**

**Instruction = 96**

**Q5.5**

**Nao tem nops, logo todas as instruções são uteis.**

**Eficiencia alta visto não ter instruções não uteis.**

**Q5.6**

**IPC = 0,873**

**Com esta arquitetura e muito difícil chegar a um IPC maior que 1. Esta arquitetura não tem Branch Predict logo vai sempre desperdiçar 2 ciclos de relogio porque o branch control esta no EX.**

**Q5.9**

**Chart, bar chart

Description automatically generated**

**Q5.10**

**Stalls = 0**

**Q5.11**

**A justificação esta no 5.6**

**Codigo Q5**

**Text

Description automatically generated with low confidence**

