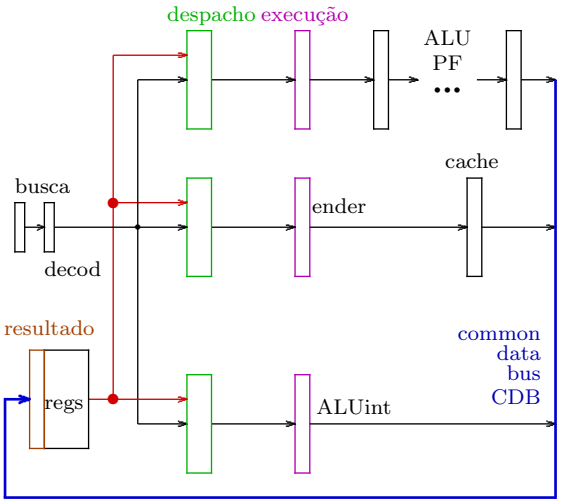
Segunda Lista de Exercícios – Escalonamento Dinâmico

O trecho de código abaixo calcula a expressão de ponto flutuante $E = A \times B + C \times D$, e os endereços de A,B,C,D,E são armazenados em r1,r2,r3,r4,r5 respectivamente. Na tabela, o acesso à memória é contado como execução.

1) Pipeline simples Calcule o número de ciclos necessários para executar este trecho de código (número de ciclos entre a emissão do primeiro LD até a emissão do último ST, inclusive) numa CPU com pipeline simples em-ordem que não tem adiantamento (diagrama ao lado). O circuito de dados inclui uma unidade de acesso à memória (LD/ST), um somador de ponto flutuante e um multiplicador de PF. Suponha que todos os LDs tem uma latência de 2 ciclos, multiplicação tem 4 ciclos e adição 2 ciclos. Escrever o resultado em registradores de PF demora 1 ciclo. Suponha também que todas as unidades funcionais são segmentadas.

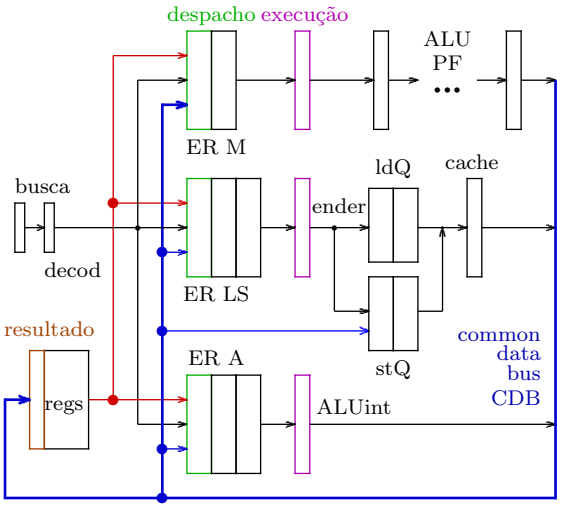


2) Escalonamento estático Reordene as instruções para minimizar o tempo de execução. Mostre a nova seqüência de instruções e calcule o número de ciclos necessários para executá-la no pipeline simples do problema 1.

3) Poucos registradores Reescreva o código usando somente *dois* registradores de PF. Otimize o código para minimizar o tempo de execução. É necessário usar variáveis temporárias em memória para manter os valores intermediários. Quando efetuado pelo compilador, este processo é chamado de *register-spilling*. Liste a nova seqüência e calcule o número de ciclos para executá-la.

4) Renomeação de registradores e escalonamento dinâmico Simule a execução do código *original* num processador com pipeline simples com renomeação de registradores e emissão fora-de-ordem. Ignore riscos estruturais que não sejam a emissão de uma instrução por ciclo. Mostre como o código é executado e calcule o número de ciclos. Compare este resultado com o do problema 2. Acrescente à tabela abaixo uma função de renomeação de registradores.

5) Efeitos de register-spilling Simule o código que você escreveu para o problema 3 no mesmo processador do problema 4 (com renomeação e execução fora-de-ordem). Compare o resultado com o do problema 3. Quais são as diferenças no programa e/ou na arquitetura que alteram o número de ciclos necessários para executar o programa? Suponha que todos os LDs antes de um ST devem ser emitidos antes que o ST seja emitido, e que LDs depois de um ST devem esperar pela emissão do ST.

instrução	ciclo decodif	entra emissão	entra execução	ciclo escrita

```
1   ld.d   f0, 0(r1)
2   ld.d   f1, 0(r2)
3   mul.d  f0, f0, f1
4   ld.d   f2, 0(r3)
5   ld.d   f3, 0(r4)
6   mul.d  f2, f2, f3
7   add.d  f0, f0, f2
8   st.d   f0, 0(r5)
```