

# Arquitetura e Organização de Computadores II

Escalonamento dinâmico

Prof. Nilton Luiz Queiroz Jr.

# Escalonamento Estático

- O escalonamento estático melhora o pipeline quando não existem dependências entre as instruções;
  - Dependências de dados que não podem ser resolvidas com forwarding causam stalls no pipeline;
- Nenhuma instrução pode ser buscada ou executada até que seja resolvida a dependência;



# Escalonamento Dinâmico

- Hardware reorganiza as instruções para reduzir a quantidade de stalls;
- Oferece diversas vantagens:
  - Códigos escritos levando em conta um certo pipeline tendem a rodar de maneira eficiente em outro pipeline;
  - Permitem manejar códigos com dependências desconhecidas;
    - Por exemplo:
      - Referenciamentos à memória;
  - Permite com que o processador tolere delays imprevisíveis;
    - Executar código esperando por dependência, por exemplo:
      - Miss em cache;



# Escalonamento Dinâmico

- As vantagens obtidas pelo escalonamento dinâmico tem um acréscimo significativo no custo do hardware;
- Tenta evitar stalls quando dependências são apresentadas;
- Arquiteturas que empregam escalonamento dinâmico podem ter seus códigos escalonados estaticamente;
  - Uma maneira de escalonar não exclui a outra;



# Problema dos pipelines simples

- Uma limitação das técnicas de pipeline simples é o fato de terem despacho execução em ordem;
  - Instruções enviadas na ordem do programa;
  - Dependências causam hazards;
    - Tendem a causar stalls;
  - Quando uma instrução  $j$  depende de uma instrução  $i$ , é necessário o fim de  $i$  para a execução de  $j$  e das posteriores;
    - Exemplo:

DIV. D	F0, F2, F4	; $i$
ADD.D	F10,F0,F8	; $j$
SUB.D	F12,F8,F14	;posterior a $j$

# Escalonamento Dinâmico

- Supondo que se deseje iniciar a execução de SUB.D:
  - É necessário que o processo seja dividido em 2 partes:
    - Verificar hazards estruturais;
    - Esperar pela ausência de hazard de dados;
  - Também é possível começar a instrução assim que seus operandos estiverem prontos;
    - Isso implicaria em execução fora de ordem;
      - Instruções executadas fora de ordem tem término fora de ordem;
    - Podem surgir hazards WAR e WAW;



# Escalonamento Dinâmico

- Para tirar proveito da execução fora de ordem é necessário múltiplas instruções no estágio de execução (EX);
  - Para isso pode-se colocar:
    - Múltiplas unidades funcionais;
    - Unidades funcionais em pipeline;
    - Combinar as duas anteriores;

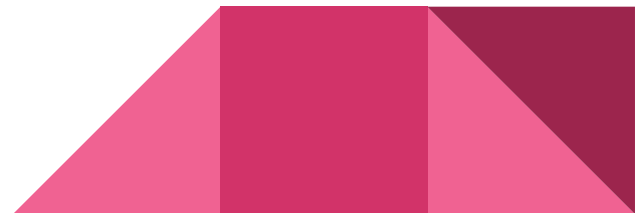


# Escalonamento Dinâmico

- Imagine o seguinte trecho de assembly:

```
DIV.D    F0, F2, F4
ADD.D    F6, F0, F8
S.D      F6, 0(R1)
SUB.D    F8, F10, F14
MUL.D    F6, F10, F8
```

- O que acontece se a instrução SUB for executada antes da instrução ADD?
- E se a instrução MUL for executada antes da instrução ADD?





# Escalonamento Dinâmico

- Imagine o seguinte trecho de assembly:

```
DIV.D    F0, F2, F4
ADD.D    F6, F0, F8
S.D      F6, 0(R1)
SUB.D    F8, F10, F14
MUL.D    F6, F10, F8
```

- O que acontece se a instrução SUB for executada antes da instrução ADD?
  - Tem-se uma antidependencia. Um hazard do tipo WAR no registrador F8;
- E se a instrução MUL for executada antes da instrução ADD?
  - Tem-se um hazard do tipo WAW no registrador F6;



# Escalonamento Dinâmico

- Imagine o seguinte trecho de assembly:

DIV.D	F0, F2, F4
ADD.D	F6, F0, F8
S.D	F6, 0(R1)
SUB.D	F8, F10, F14
MUL.D	F6, F10, F8

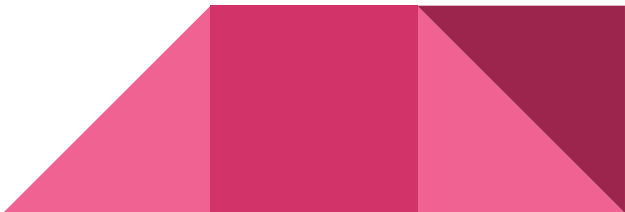
- Observe que também existem 3 dependências de dados verdadeiras. Quais são elas?



# Escalonamento Dinâmico

- Imagine o seguinte trecho de assembly:

```
DIV.D    F0, F2, F4
ADD.D    F6, F0, F8
S.D      F6, 0(R1)
SUB.D    F8, F10, F14
MUL.D    F6, F10, F8
```

- Observe que também existem 3 dependências de dados verdadeiras. Quais são elas?
    - DIV.D e ADD.D: registrador F0;
    - ADD.D e S.D: registrador F6;
    - SUB.D e MUL.D: registrador F8
- 

# Escalonamento Dinâmico

- O que poderia ser feito para evitar esse tipo de hazard introduzido pela execução das instruções fora de ordem?

DIV.D	F0, F2, F4
ADD.D	S, F0, F8
S.D	F6, 0(R1)
SUB.D	T, F10, F14
MUL.D	F6, F10, F8



# Escalonamento Dinâmico

- O que poderia ser feito para evitar esse tipo de hazard introduzido pela execução das instruções fora de ordem?
  - Renomear Registradores

DIV.D	F0, F2, F4
ADD.D	S, F0, F8
S.D	F6, 0(R1)
SUB.D	T, F10, F14
MUL.D	F6, F10, F8



# Renomeação de Registradores

- Suponha novamente o seguinte trecho de código:

```
DIV.D    F0, F2, F4
ADD.D    F6, F0, F8
S.D      F6, 0(R1)
SUB.D    F8, F10, F14
MUL.D    F6, F10, F8
```

- Como ficaria o código com os registradores renomeado



# Renomeação de registradores

- Suponha a execução em ordem e fora de ordem dos dois códigos.
  - Suponha que inicialmente : F2=6; F4=2; F8=2; F10=4;F14=1;
    - Na renomeação assuma o valor inicial para T = 2 e desconsidere o valor inicial de S;
  - Imagine que a execução fora de ordem seja: 1, 4, 5 2, 3

1.	DIV.D	F0, F2, F4
2.	ADD.D	S, F0, F8
3.	S.D	F6, 0(R1)
4.	SUB.D	T, F10, F14
5.	MUL.D	F6, F10, F8

1.	DIV.D	F0, F2, F4
2.	ADD.D	F6, F0, F8
3.	S.D	F6, 0(R1)
4.	SUB.D	F8, F10, F14
5.	MUL.D	F6, F10, F8

# Renomeação de registradores

	Ordem	Fora de ordem	Renom. Reg.
F0	3	3	3
F2	6	6	6
F4	2	2	2
F6	12	6	12
F8	3	3	3
F10	4	4	4
F14	1	1	1
M[R1]	5	6	12
S	-	-	6
t	-	-	2



# Renomeação de registradores

- O resultado final gerado é errado para ambos os casos
  - Apenas executar fora de ordem;
  - Renomear registradores
- Como consertar a renomeação de registradores?



# Renomeação de registradores

- O resultado final gerado é errado para ambos os casos
  - Apenas executar fora de ordem;
  - Renomear registradores
- Como consertar a renomeação de registradores?
  - Propagar registradores renomeados para dependências reais.



# Renomeação de registradores

- Renomear registradores;

1. DIV.D      F0, F2, F4
2. ADD.D      F6, F0, F8
3. S.D        F6, 0(R1)
4. SUB.D      F8, F10, F14
5. MUL.D      F6, F10, F8



# Renomeação de registradores

- Renomear registradores;

1.	DIV.D	F0, F2, F4
2.	ADD.D	F6, F0, F8
3.	S.D	F6, 0(R1)
4.	SUB.D	F8, F10, F14
5.	MUL.D	F6, F10, F8

1.	DIV.D	F0, F2, F4
2.	ADD.D	S, F0, F8
3.	S.D	S, 0(R1)
4.	SUB.D	T, F10, F14
5.	MUL.D	F6, F10, T

- Como ficaria a execução do código com os registradores renomeados
  - Suponha que inicialmente : F2=6; F4=2; F8=2; F10=4;F14=1;
  - Para execução fora de ordem execute na ordem 1,4,5,2,3 com T = 2

# Renomeação de registradores

	Ordem	Reg. Renom.
F0	3	3
F2	6	6
F4	2	2
F6	12	12
F8	3	2
F10	4	4
F14	1	1
M[R1]	5	5
S	-	5
t	-	3



# Escalonamento Dinâmico

- O término de execução fora de ordem também cria complicações em tratamento de exceções;
  - As exceções devem ser exatamente as mesmas quando o programa é executado em ordem ou fora de ordem para entradas iguais;



# Escalonamento Dinâmico

- Processadores escalonados dinamicamente podem gerar exceções imprecisas;
  - Ocorrem exceções imprecisas quando o estado do processador no momento que a exceção foi gerada não é o mesmo que seria caso a execução fosse sequencial;
- As exceções imprecisas ocorrem por dois motivos:
  - Instruções já completadas que estão adiante na ordem do programa que a instrução que causa exceção;
  - O pipeline ainda não ter completado instruções que estão atrás na ordem do programa que a instrução que causa a exceção;;



# Escalonamento Dinâmico

- Para permitir a execução fora de ordem é necessário dividir o estágio de decodificação em dois estágios:
  - Despacho;
    - Decodificar a instrução e verificar hazards estruturais;
  - Leitura de operandos;
    - Esperar até que não haja hazards de dados e depois ler operandos;
- O estágio de busca de instrução precede o estágio de despacho;
  - Instruções buscadas podem ser colocadas em um registrador de instruções ou em uma fila de instruções pendentes
- O estágio de execução vem após o estágio de leitura de operandos;
  - A execução pode levar diversos ciclos;
    - Depende da instrução;



# Escalonamento Dinâmico

- Instruções passam pelo estágio de despacho em ordem;
  - Podem ser reordenadas no segundo estágio (busca de operadores);
- Existem diferentes técnicas para permitir execução fora de ordem:
  - Scoreboarding;
  - Algoritmo de Tomasulo;
  - Algoritmo de Tomasulo estendido para especulação;



# Escalonamento Dinâmico

- Algoritmo de Tomasulo:
  - Inventado por Robert Tomasulo;
  - Verifica quando os operandos para as instruções estão disponíveis;
    - Minimizar hazards RAW;
  - Renomeia registradores;
    - Minimiza hazards WAW e WAR;
- O algoritmo a seguir enfoca a unidade de ponto flutuante e a unidade de load-store do conjunto de instruções MIPS;
  - É também levado em conta que tem-se disponível várias unidades funcionais;



# Algoritmo de Tomasulo

- Renomeação de registradores:
  - Elimina hazards WAR e WAW
  - Renomeia todos os registradores destino das instruções que causam esses hazards;
  - Faz uso de estações de reserva;
    - Faz buffer de operandos das instruções sempre que eles estão disponíveis;
  - Instruções pendentes designam as estações que irão prover sua entrada;
  - Quando a escrita for feita no registrador, somente a última de fato atualiza o registrador;
  - Conforme as instruções são despachadas os registradores com pendências são renomeados para os nomes das estações de reserva;
  - Essa técnica pode eliminar hazards que não seriam possíveis para um compilador;
    - Pois existem mais estações de reservas do que registradores de fato;

# Reserva de Registradores

- A reserva de registradores distribui a detecção de hazards de controle e também a execução;
  - Cada estação de reserva sabe quando deve começar uma nova execução na unidade funcional a qual ela pertence;
- O resultados são passados diretamente para as unidades funcionais a partir das estações de reserva;
  - Lá eles são mantidos em buffer em vez de passarem pelos registradores;
- É necessário um barramento para que todas as unidades que esperam por um operando sejam carregadas simultaneamente (barramento de dados comum);
  - Quando existem mais de uma unidade de execução que recebem várias instruções por clock são necessários diversos barramentos;

# Reserva de Registradores

- Todas as estações de reserva usam tags para o controle do pipeline;
- São necessários store buffers (buffers de armazenamento) e load buffers (buffers de carregamento);
  - Esse buffers mantém os dados ou endereços indo e vindo da memória;
    - Funcionam de maneira parecida as estações de reserva;
- Os registradores de ponto flutuante são conectados por um par de barramentos para unidades funcionais, e um único barramento para unidades de load-store;



# Caminho da instrução

- Podemos dividir as etapas que uma instrução passa em 3 principais:
  - Despacho;
  - Execução;
  - Escrita;
- Cada uma delas pode levar um número arbitrário de ciclos;



# Caminho da instrução

- Despacho:
  - Carrega a nova instrução da fila de instrução;
  - Caso exista estação de reserva vazia então envie-a para ela;
    - Com os valores de operando, se estiverem nos registradores;
  - Se não houver uma estação vazia tem-se um hazard estrutural (o mesmo para buffers);
    - Stall no pipeline até que libere uma estação ou buffer;
  - Se os operandos não estiverem nos registradores continue acompanhando as unidades funcionais que produzirão os operandos;
  - Essa etapa é responsável por renomear os registradores;



# Caminho da instrução

- Execução:
  - Quando um ou mais operandos não estão disponíveis deve-se monitorar o barramento comum enquanto se aguarda o fim de sua execução;
  - Quando um operando fica disponível ele é colocado em uma estação de reserva;
    - Quando todos estão prontos a operação está pronta para ser executada;
      - Se existem várias instruções prontas para executar escolher uma arbitrariamente;
        - Loads e stores precisam de das etapas:
          - Cálculo do endereço efetivo;
          - Acesso a memória;
            - Loads precisam da unidade de memória disponível;
            - Stores precisam esperar o valor a ser armazenado;



# Caminho da instrução

- Reduz atrasos por conflitos RAW;
- Nenhuma instrução pode iniciar a execução até a conclusão de todos desvios posteriores;
  - Garante que exceções sejam executadas;



# Caminho da instrução

- Escrita do resultado:
  - Quando o resultado estiver disponível deve ser escrito no barramento de dados comum;
    - Uma vez escritos no barramento é então propagado para os registradores;
  - Stores são mantidos no buffer de stores até que o valor a ser armazenado e o endereço para escrita estejam disponíveis;
    - Resultado é escrito assim que a unidade de memória ficar livre;



# Caminho da instrução

- As estruturas que detectam hazards devem ser conectadas as:
  - Estações de reserva;
  - Ao banco de registradores; e
  - Aos buffers de load e store;
- Operações enviadas que aguardam operandos se referem a eles pelo número da estação de reserva atribuída a qual o operando será escrito;
  - Valores não usados, como 0, indicam que o operando está disponível nos registradores;



# Caminho da instrução

- Os resultados são transmitidos por broadcast a um barramento que é monitorado pelas estações de reservas;
- A combinação desse barramento com recuperação de resultados do barramento pelas estações de reserva implementa o mecanismo de bypass e forwarding;
  - Isso introduz um ciclo de latência entre a fonte e o resultado;
- Vale lembrar que as tags no esquema de Tomasulo se referem ao buffer ou a unidade que vai produzir o resultado;



# Estações de reserva

- Cada estação de reserva possui sete campos:
  - Op: operação a ser realizada sobre os operandos fonte S1 e S2;
  - Qj e Qk: As estações de reserva que produzirão os operandos-fonte correspondentes
    - Zero indica que os operandos estão prontos ou são desnecessários;
  - Vj e Vk: Valor dos operandos fontes;
  - A: Usado para manter informações utilizadas no cálculo de memória para loads ou stores;
    - Inicialmente o campo imediato da instrução é armazenado no campo A;
    - Após o cálculo de endereço, o endereço efetivo que é armazenado;
  - Busy: indica se a estação de reserva ou “sua” unidade funcional estão ocupados;



# Banco de registradores

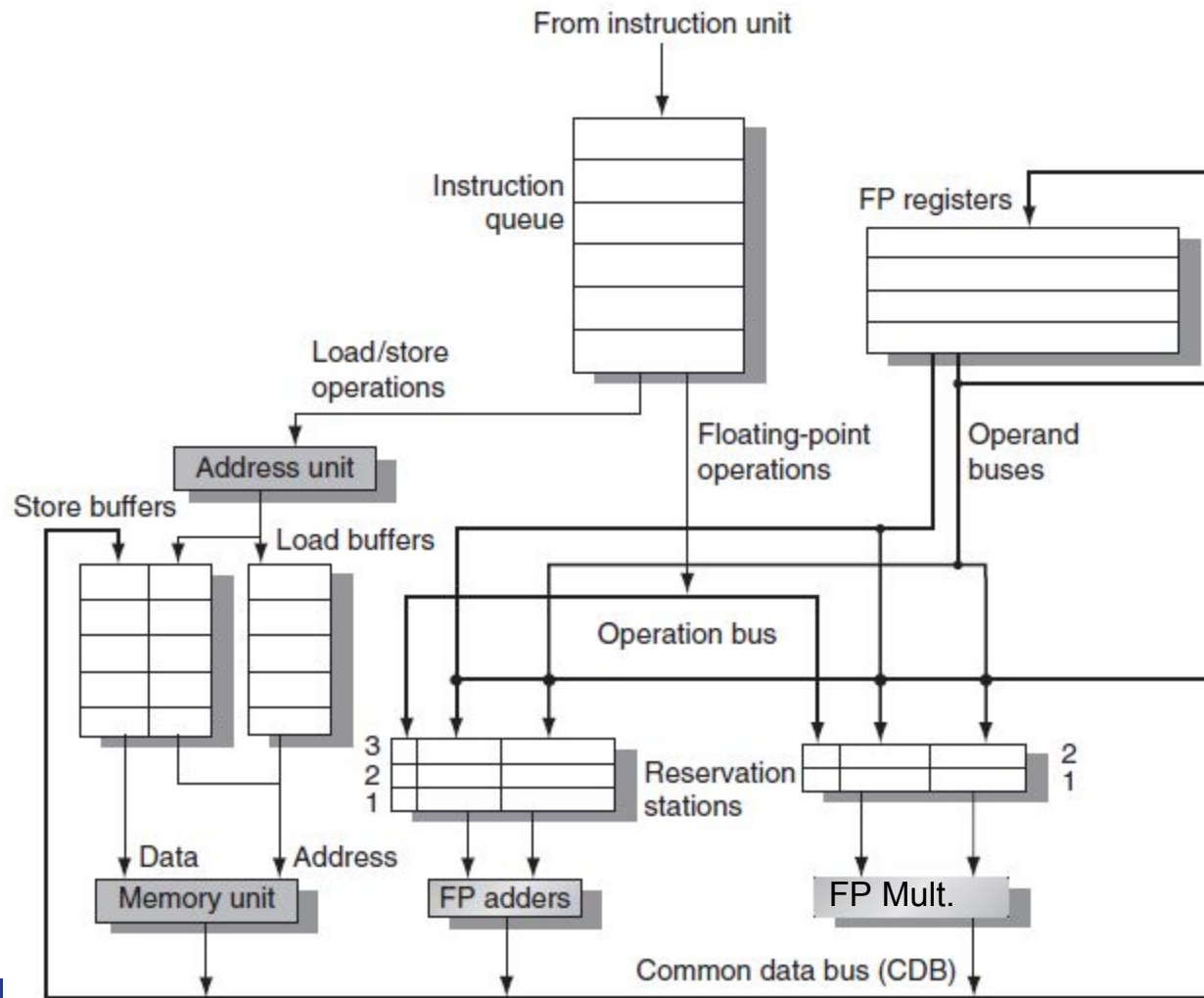
- O banco de registradores possui um campo  $Q_i$ :
  - Guarda o número da estação de reserva que contém a operação cujo o resultado será armazenado nesse registrador;
    - Quando  $Q_i = 0$  não existe instrução ativa calculando um valor que futuramente será armazenado nesse registrador, isso significa que o valor é simplesmente o conteúdo do registrador;



# Buffers de Load e Store

- Buffers de Load e Store possuem um campo A:
  - Mantém o resultado do endereço efetivo quando a primeira etapa da execução da instrução (cálculo do endereço) tiver sido concluída;







# Algoritmo de Tomasulo

- Para o algoritmo assuma:
  - *rs* e *rt*: Os registradores fonte;
  - *rd*: O registrador destino;
  - *r*: A estação de reserva (ou buffer) relacionado com a instrução;
  - *result*: o valor retornado por uma unidade funcional;
  - RS: A estrutura de dados que representa as estações de reserva;
  - RegisterStat: Estrutura de dados que mantém os estados dos registradores;
    - Note que RegisterStat não são os registradores em si.
  - Reg: Estrutura que mantém os valores dos registradores;



# Algoritmo de Tomasulo - Operações UFP

Despacho:

```
if (RegisterStat[rs].Qi != 0){
    RS[r].Qj ← RegisterStat[rs].Qi
}else{
    RS[r].Vj ← Regs[rs];
    RS[r].Qj ← 0;
}
if (RegisterStat[rt].Qi != 0){
    RS[r].Qk ← RegisterStat[rt].Qi
}else{
    RS[r].Vk ← Regs[rt];
    RS[r].Qk ← 0;
}
RS[r].Busy ← yes;
RegisterStat[rd].Q ← r;
```

Execução:

```
result ← RS[r].Vj `OP` RS[r].Vk
```

Escrita:

```
∀ x (
    if (RegisterStat[x].Qi=r){
        Regs[x] ← result;
        RegisterStat[x].Qi ← 0;
    };
)
∀ x(
    if (RS[x].Qj=r){
        RS[x].Vj ← result;RS[x].Qj ← 0
    }
);
∀ x(
    if (RS[x].Qk=r) {
        RS[x].Vk ← result;RS[x].Qk ← 0
    }
);
RS[r].Busy ← no;
```

# Algoritmo de Tomasulo - Load

Despacho:

```
if (RegisterStat[rs].Qi != 0){  
    RS[r].Qj ← RegisterStat[rs].Qi;  
}else {  
    RS[r].Vj ← Regs[rs];  
    RS[r].Qj ← 0;  
}  
RS[r].A ← imm;  
RS[r].Busy ← yes;  
RegisterStat[r].Qi ← r;
```

Execução:

```
RS[r].A ← RS[r].Vj + RS[r].A;
```

```
result ← Mem[RS[r].A];
```

Escrita:

```
∀ x (  
    if (RegisterStat[x].Qi=r){  
        Regs[x] ← result;  
        RegisterStat[x].Qi ← 0;  
    };  
)  
∀ x(  
    if (RS[x].Qj=r){  
        RS[x].Vj ← result; RS[x].Qj ← 0  
    }  
);  
∀ x(  
    if (RS[x].Qk=r) {  
        RS[x].Vk ← result; RS[x].Qk ← 0  
    }  
);  
RS[r].Busy ← no;
```

# Algoritmo de Tomasulo - Store

Despacho:

```
if (RegisterStat[rs].Qi != 0){  
    RS[r].Qj ← RegisterStat[rs].Qi;  
}else {  
    RS[r].Vj ← Regs[rs];  
    RS[r].Qj ← 0;  
}  
RS[r].A ← imm;  
RS[r].Busy ← yes;
```

```
if (RegisterStat[rt].Qi != 0){  
    RS[r].Qk ← RegisterStat[rs].Qi;  
}else {  
    RS[r].Vk ← Regs[rt];  
    RS[r].Qk ← 0;  
}
```

Execução:

```
RS[r].A ← RS[r].Vj + RS[r].A;
```

Escrita:

```
Mem[RS[r].A] ← RS[r].Vk;  
RS[r].Busy ← no;
```



# Algoritmo de Tomasulo

- Antes de cada etapa é necessário que algumas condições sejam satisfeitas:
  - Operações UFP:
    - Despacho:
      - Aguardar até que a estação de reserva tenha locais;
    - Execução:
      - Unidade Funcional livre;
      - Aguardar  $Q_j$  e  $Q_k$  indicando valores prontos
        - $RS[r].Q_k = 0$  e  $RS[r].Q_j = 0$ ;
    - Escrita de resultado:
      - Aguardar execução completa;
      - Barramento de dados (CDB) disponível;



# Algoritmo de Tomasulo

- Loads:
  - Despacho:
    - Aguardar até que o buffer de reserva tenha locais;
  - Execução:
  - Unidade Funcional livre;
    - Aguardar Qj indicando valor pronto
      - $RS[r].Qj = 0$ ;
    - A estação de reserva ser a primeira da fila de load store;
      - Loads e stores devem ser feitos em ordem!
    - A primeira etapa da instrução de load deve ter sido completada;
      - O endereço de leitura deve ter sido calculado
        - $RS[r].A \leftarrow RS[r].Vj + RS[r].A$ ;
  - Escrita de resultado:
    - Aguardar execução completa;
    - Barramento de dados (CDB) disponível;

# Algoritmo de Tomasulo

- Stores:
  - Despacho:
    - Aguardar até que o buffer de reserva tenha locais;
  - Execução:
  - Unidade Funcional livre;
    - Aguardar  $Q_j$  indicando valor pronto
      - $RS[r].Q_j = 0$ ;
    - A estação de reserva ser a primeira da fila de load store;
  - Escrita de resultado:
    - Aguardar execução completa;
    - Aguardar  $Q_k$  indicando valor pronto;
      - $RS[r].Q_k = 0$ ;



# Exemplo de execução



# Execução

- Para a execução é necessário conhecer alguns detalhes do hardware:
  - Quantidade de unidades funcionais;
  - Quantidade de espaços em cada estação de reserva;
  - Ciclos necessários para cada unidade funcional;
- Neste exemplo iremos adotar:
  - 3 estações de reserva para operações add e sub;
  - 2 para operações mul e div;
  - Uma unidade funcional para soma e uma para multiplicação
    - A de soma resolve também subtrações e a de multiplicação resolve divisões
  - Somas, subtrações e loads levam 2 ciclos na execução;
  - Multiplicações levam 10 ciclos;
  - divisões levam 40 ciclos;
  - Assuma que as operações não podem começar no mesmo ciclo que sua dependência foi resolvida (devem começar no ciclo da escrita);

[illegible]

Status das instruções					Execução	Escrita do						
Instrução		<i>j</i>	<i>k</i>	Despacho	Completa	Resultado						
LD	F6	34+	R2	1								
LD	F2	45+	R3									
MULTD	F0	F2	F4									
SUBD	F8	F6	F2									
DIVD	F10	F0	F6									
ADDD	F6	F8	F2									
Estações de reserva					S1	S2	RS para <i>j</i>	RS para <i>k</i>	A			
	Tempo	Nome	Busy	Op	V <sub>j</sub>	V <sub>k</sub>	Q <sub>j</sub>	Q <sub>k</sub>				
	0	Add1	Não									
	0	Add2	Não									
	0	Add3	Não									
	0	Mult1	Não									
	0	Mult2	Não									
	2	Load1	Sim	Load					34+R2			
	0	Load2	Não									
	0	Load3	Não									
Status dos registradores												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
1		Q <sub>i</sub>					Load1					

Status das instruções					Execução	Escrita do						
Instrução		<i>j</i>	<i>k</i>	Despacho	Completa	Resultado						
LD	F6	34+	R2	1								
LD	F2	45+	R3	2								
MULTD	F0	F2	F4									
SUBD	F8	F6	F2									
DIVD	F10	F0	F6									
ADDD	F6	F8	F2									
Estações de reserva					S1	S2	RS para <i>j</i>	RS para <i>k</i>	A			
	Tempo	Nome	Busy	Op	V <sub>j</sub>	V <sub>k</sub>	Q <sub>j</sub>	Q <sub>k</sub>				
	0	Add1	Não									
	0	Add2	Não									
	0	Add3	Não									
	0	Mult1	Não									
	0	Mult2	Não									
	1	Load1	Sim	Load					34+R2			
	2	Load2	Sim	Load					45+R3			
	0	Load3	Não									
Status dos registradores												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
2			Q <sub>i</sub>		Load2		Load1					

Status das instruções					Execução	Escrita do
Instrução		<i>j</i>	<i>k</i>	Despacho	Completa	Resultado
LD	F6	34+	R2	1	3	
LD	F2	45+	R3	2		
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

Note que Qj “aponta” para o valor que será obtido na operação armazenada em Load2. Assim, a operação MULTD não depende mais de F2.

Estações de reserva					S1	S2	RS para <i>j</i>	RS para <i>k</i>	A
	Tempo	Nome	Busy	Op	Vj	Vk	Qj	Qk	
	0	Add1	Não						
	0	Add2	Não						
	0	Add3	Não						
	0	Mult1	SIM	MULTD		Valor (F4)	Load2		
	0	Mult2	Não						
	0	Load1	Sim	Load					34+R2
	1	Load2	Sim	Load					45+R3
	0	Load3	Não						

#### Status dos registradores

Clock			F0	F2	F4	F6	F8	F10	F12	...	F30
3		Qi	Mult1	Load2		Load1					

O valor 0 indica que o registrador está pronto para uso

Status das instruções					Execução	Escrita do				
Instrução		<i>j</i>	<i>k</i>	Despacho	Completa	Resultado				
LD	F6	34+	R2	1	3	4				
LD	F2	45+	R3	2	4					
MULTD	F0	F2	F4	3						
SUBD	F8	F6	F2	4						
DIVD	F10	F0	F6							
ADDD	F6	F8	F2							
Estações de reserva					S1	S2	RS para <i>j</i>	RS para <i>k</i>	A	
	Tempo	Nome	Busy	Op	V <sub>j</sub>	V <sub>k</sub>	Q <sub>j</sub>	Q <sub>k</sub>		
	0	Add1	Sim	SUBD	M[34+R2]			Load2		
	0	Add2	Não							
	0	Add3	Não							
	0	Mult1	Sim	MULTD		Valor (F4)	Load2			
	0	Mult2	Não							
	0	Load1	Não							
	0	Load2	Sim	Load					45+R3	
	0	Load3	Não							
Status dos registradores										
Clock				F0	F2	F4	F6	F8	F10	F12 ... F30
4		Qi		Mult1	Load2			0 Add1		

Status das instruções				Execução		Escrita do							
Instrução		<i>j</i>	<i>k</i>	Despacho	Completa	Resultado							
LD	F6	34+	R2	1	3	4							
LD	F2	45+	R3	2	4	5							
MULTD	F0	F2	F4	3									
SUBD	F8	F6	F2	4									
DIVD	F10	F0	F6	5									
ADDD	F6	F8	F2										
Estações de reserva					S1	S2	RS para <i>j</i>	RS para <i>k</i>	A				
	Tempo	Nome	Busy	Op	V <sub>j</sub>	V <sub>k</sub>	Q <sub>j</sub>	Q <sub>k</sub>					
	2	Add1	Sim	SUBD	M[34+R2]	M[45+R3]							
	0	Add2	Não										
	0	Add3	Não										
	10	Mult1	Sim	MULTD	M[45+R3]	Valor (F4)							
	0	Mult2	Sim	DIVD		M[34+R2]	Mult1						
	0	Load1	Não										
	0	Load2	Não										
	0	Load3	Não										
Status dos registradores													
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30	
5		Qi	Mult1		0		0	Add1	Mult2				



Status das instruções					Execução	Escrita do						
Instrução		<i>j</i>	<i>k</i>	Despacho	Completa	Resultado						
LD	F6	34+	R2	1	3	4						
LD	F2	45+	R3	2	4	5						
MULTD	F0	F2	F4	3								
SUBD	F8	F6	F2	4								
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6								
Estações de reserva					S1	S2	RS para <i>j</i>	RS para <i>k</i>	A			
	Tempo	Nome	Busy	Op	V <sub>j</sub>	V <sub>k</sub>	Q <sub>j</sub>	Q <sub>k</sub>				
	1	Add1	Sim	SUBD	M[34+R2]	M[45+R3]						
	0	Add2	Sim	ADDD		M[45+R3]	Add1					
	0	Add3	Não									
	9	Mult1	Sim	MULTD	M[45+R3]	Valor (F4)						
	0	Mult2	Sim	DIVD		M[34+R2]	Mult1					
	0	Load1	Não									
	0	Load2	Não									
	0	Load3	Não									
Status dos registradores												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
6			Qi	Mult1	0		Add2	Add1	Mult2			



Status das instruções

Instrução		<i>j</i>	<i>k</i>	Despacho	Execução Completa	Escrita do Resultado						
LD	F6	34+	R2	1	3	4						
LD	F2	45+	R3	2	4	5						
MULTD	F0	F2	F4	3								
SUBD	F8	F6	F2	4	7							
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6								

Estações de reserva

					S1	S2	RS para <i>j</i>	RS para <i>k</i>	A
	<i>Tempo</i>	<i>Nome</i>	<i>Busy</i>	<i>Op</i>	<i>V<sub>j</sub></i>	<i>V<sub>k</sub></i>	<i>Q<sub>j</sub></i>	<i>Q<sub>k</sub></i>	
	0	Add1	Sim	SUBD	M[34+R2]	M[45+R3]			
	0	Add2	Sim	ADDD		M[45+R3]	Add1		
	0	Add3	Não						
	8	Mult1	Sim	MULTD	M[45+R3]	Valor (F4)			
	0	Mult2	Sim	DIVD		M[34+R2]	Mult1		
	0	Load1	Não						
	0	Load2	Não						
	0	Load3	Não						

Status dos registradores

Clock			F0	F2	F4	F6	F8	F10	F12	...	F30
7		Qi	Mult1	0		Add2	Add1	Mult2			

Status das instruções				Execução		Escrita do					
Instrução		<i>j</i>	<i>k</i>	Despacho	Completa	Resultado					
LD	F6	34+	R2	1	3	4					
LD	F2	45+	R3	2	4	5					
MULTD	F0	F2	F4	3							
SUBD	F8	F6	F2	4	7	8					
DIVD	F10	F0	F6	5							
ADDD	F6	F8	F2	6							
Estações de reserva					S1	S2	RS para <i>j</i>	RS para <i>k</i>	A		
	Tempo	Nome	Busy	Op	V <sub>j</sub>	V <sub>k</sub>	Q <sub>j</sub>	Q <sub>k</sub>			
	0	Add1	Não								
	2	Add2	Sim	ADDD	M[34+R2]-M[45+R3]	M[45+R3]					
	0	Add3	Não								
	7	Mult1	Sim	MULTD	M[45+R3]	Valor (F4)					
	0	Mult2	Sim	DIVD		M[34+R2]	Mult1				
	0	Load1	Não								
	0	Load2	Não								
	0	Load3	Não								
Status dos registradores											
Clock				F0	F2	F4	F6	F8	F10	F12	... F30
8		Qi	Mult1		0		Add2		0	Mult2	

Status das instruções				Execução		Escrita do							
Instrução		<i>j</i>	<i>k</i>	Despacho	Completa	Resultado							
LD	F6	34+	R2	1	3	4							
LD	F2	45+	R3	2	4	5							
MULTD	F0	F2	F4	3									
SUBD	F8	F6	F2	4	7	8							
DIVD	F10	F0	F6	5									
ADDD	F6	F8	F2	6									
Estações de reserva				S1		S2	RS para <i>j</i>	RS para <i>k</i>	A				
	Tempo	Nome	Busy	Op	V <sub>j</sub>	V <sub>k</sub>	Q <sub>j</sub>	Q <sub>k</sub>					
	0	Add1	Não										
	1	Add2	Sim	ADDD	M[34+R2]-M[45+R3]	M[45+R3]							
	0	Add3	Não										
	6	Mult1	Sim	MULTD	M[45+R3]	Valor (F4)							
	0	Mult2	Sim	DIVD		M[34+R2]	Mult1						
	0	Load1	Não										
	0	Load2	Não										
	0	Load3	Não										
Status dos registradores													
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30	
9		Qi	Mult1		0		Add2		0	Mult2			

Status das instruções					Execução	Escrita do						
Instrução		<i>j</i>	<i>k</i>	Despacho	Completa	Resultado						
LD	F6	34+	R2	1	3	4						
LD	F2	45+	R3	2	4	5						
MULTD	F0	F2	F4	3								
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10							
Estações de reserva					S1	S2	RS para <i>j</i>	RS para <i>k</i>	A			
	Tempo	Nome	Busy	Op	V <sub>j</sub>	V <sub>k</sub>	Q <sub>j</sub>	Q <sub>k</sub>				
	0	Add1	Não									
	0	Add2	Não									
	0	Add3	Não									
	5	Mult1	Sim	MULTD	M[45+R3]	Valor (F4)						
	0	Mult2	Sim	DIVD		M[34+R2]	Mult1					
	0	Load1	Não									
	0	Load2	Não									
	0	Load3	Não									
Status dos registradores												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
10			Qi	Mult1	0		Add2		0 Mult2			

Status das instruções					Execução	Escrita do						
Instrução		<i>j</i>	<i>k</i>	Despacho	Completa	Resultado						
LD	F6	34+	R2	1	3	4						
LD	F2	45+	R3	2	4	5						
MULTD	F0	F2	F4	3								
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10	11						
Estações de reserva					S1	S2	RS para <i>j</i>	RS para <i>k</i>	A			
	Tempo	Nome	Busy	Op	V <sub>j</sub>	V <sub>k</sub>	Q <sub>j</sub>	Q <sub>k</sub>				
	0	Add1	Não									
	0	Add2	Não									
	0	Add3	Não									
	4	Mult1	Sim	MULTD	M[45+R3]	Valor (F4)						
	0	Mult2	Sim	DIVD		M[34+R2]	Mult1					
	0	Load1	Não									
	0	Load2	Não									
	0	Load3	Não									
Status dos registradores												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
11			Q <sub>i</sub>	Mult1	0		0	0	Mult2			

Status das instruções					Execução	Escita do						
Instrução		<i>j</i>	<i>k</i>	Despacho	Completa	Resultado						
LD	F6	34+	R2	1	3	4						
LD	F2	45+	R3	2	4	5						
MULTD	F0	F2	F4	3								
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADD	F6	F8	F2	6	10	11						
Estações de reserva					S1	S2	RS para <i>j</i>	RS para <i>k</i>	A			
	Tempo	Nome	Busy	Op	V <sub>j</sub>	V <sub>k</sub>	Q <sub>j</sub>	Q <sub>k</sub>				
	0	Add1	Não									
	0	Add2	Não									
	0	Add3	Não									
	3	Mult1	Sim	MULTD	M[45+R3]	Valor (F4)						
	0	Mult2	Sim	DIVD		M[34+R2]	Mult1					
	0	Load1	Não									
	0	Load2	Não									
	0	Load3	Não									
Status dos registradores												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
12			Qi	Mult1	0		0		0	Mult2		



Status das instruções

Instrução		<i>j</i>	<i>k</i>	<i>Despacho</i>	<i>Execução Completa</i>	<i>Escrita do Resultado</i>						
LD	F6	34+	R2	1	3	4						
LD	F2	45+	R3	2	4	5						
MULTD	F0	F2	F4	3								
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10	11						

Estações de reserva

	<i>Tempo</i>	<i>Nome</i>	<i>Busy</i>	<i>Op</i>	<i>S1</i> <i>Vj</i>	<i>S2</i> <i>Vk</i>	<i>RS para j</i> <i>Qj</i>	<i>RS para k</i> <i>Qk</i>	<i>A</i>
	0	Add1	Não						
	0	Add2	Não						
	0	Add3	Não						
	2	Mult1	Sim	MULTD	M[45+R3]	Valor (F4)			
	0	Mult2	Sim	DIVD		M[34+R2]	Mult1		
	0	Load1	Não						
	0	Load2	Não						
	0	Load3	Não						

Status dos registradores

Clock			<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
13		<i>Qi</i>	Mult1	0		0		0 Mult2			

Status das instruções					Execução	Escrita do						
Instrução		<i>j</i>	<i>k</i>	Despacho	Completa	Resultado						
LD	F6	34+	R2	1	3	4						
LD	F2	45+	R3	2	4	5						
MULTD	F0	F2	F4	3								
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10	11						
Estações de reserva					S1	S2	RS para <i>j</i>	RS para <i>k</i>	A			
	Tempo	Nome	Busy	Op	V <sub>j</sub>	V <sub>k</sub>	Q <sub>j</sub>	Q <sub>k</sub>				
	0	Add1	Não									
	0	Add2	Não									
	0	Add3	Não									
	1	Mult1	Sim	MULTD	M[45+R3]	Valor (F4)						
	0	Mult2	Sim	DIVD		M[34+R2]	Mult1					
	0	Load1	Não									
	0	Load2	Não									
	0	Load3	Não									
Status dos registradores												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
14			Qi	Mult1	0		0		0	Mult2		



Status das instruções					Execução	Escrita do						
Instrução		<i>j</i>	<i>k</i>	Despacho	Completa	Resultado						
LD	F6	34+	R2	1	3	4						
LD	F2	45+	R3	2	4	5						
MULTD	F0	F2	F4	3	15							
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10	11						
Estações de reserva					S1	S2	RS para <i>j</i>	RS para <i>k</i>	A			
	Tempo	Nome	Busy	Op	V <sub>j</sub>	V <sub>k</sub>	Q <sub>j</sub>	Q <sub>k</sub>				
	0	Add1	Não									
	0	Add2	Não									
	0	Add3	Não									
	0	Mult1	Sim	MULTD	M[45+R3]	Valor (F4)						
	0	Mult2	Sim	DIVD		M[34+R2]	Mult1					
	0	Load1	Não									
	0	Load2	Não									
	0	Load3	Não									
Status dos registradores												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
15		Q <sub>i</sub>	Mult1		0		0		0	Mult2		

Status das instruções				Execução		Escrita do									
Instrução		j	k	Despacho	Completa	Resultado									
LD	F6	34+	R2	1	3	4									
LD	F2	45+	R3	2	4	5									
MULTD	F0	F2	F4	3	15	16									
SUBD	F8	F6	F2	4	7	8									
DIVD	F10	F0	F6	5											
ADDD	F6	F8	F2	6	10	11									
Estações de reserva					S1	S2	RS para j	RS para k	A						
	Tempo	Nome	Busy	Op	Vj	Vk	Qj	Qk							
	0	Add1	Não												
	0	Add2	Não												
	0	Add3	Não												
	0	Mult1	Não												
	40	Mult2	Sim	DIVD	M[45+R3]*F4	M[34+R2]									
	0	Load1	Não												
	0	Load2	Não												
	0	Load3	Não												
Status dos registradores															
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30			
16		Qi		0	0		0		0	Mult2					

39 ciclos depois



Status das instruções

Instrução		<i>j</i>	<i>k</i>	Despacho	Execução Completa	Escrita do Resultado
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

Estações de reserva

				<i>S1</i>	<i>S2</i>	<i>RS para j</i>	<i>RS para k</i>	<i>A</i>
	<i>Tempo</i>	<i>Nome</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>
	0	Add1	Não					
	0	Add2	Não					
	0	Add3	Não					
	0	Mult1	Não					
	1	Mult2	Sim	DIVD	M[45+R3]*F4	M[34+R2]		
	0	Load1	Não					
	0	Load2	Não					
	0	Load3	Não					

Status dos registradores

Clock			<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
55		<i>Qi</i>	0	0		0		0 Mult2			

Status das instruções					Execução	Escrita do						
Instrução		<i>j</i>	<i>k</i>	Despacho	Completa	Resultado						
LD	F6	34+	R2	1	3	4						
LD	F2	45+	R3	2	4	5						
MULTD	F0	F2	F4	3	15	16						
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5	56							
ADDD	F6	F8	F2	6	10	11						
Estações de reserva					S1	S2	RS para <i>j</i>	RS para <i>k</i>	A			
	Tempo	Nome	Busy	Op	V <sub>j</sub>	V <sub>k</sub>	Q <sub>j</sub>	Q <sub>k</sub>				
	0	Add1	Não									
	0	Add2	Não									
	0	Add3	Não									
	0	Mult1	Não									
	0	Mult2	Sim	DIVD	M[45+R3]*F4	M[34+R2]						
	0	Load1	Não									
	0	Load2	Não									
	0	Load3	Não									
Status dos registradores												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
56			Qi	0	0		0		0 Mult2			

Status das instruções				Execução		Escita do							
Instrução		j	k	Despacho	Completa	Resultado							
LD	F6	34+	R2	1	3	4							
LD	F2	45+	R3	2	4	5							
MULTD	F0	F2	F4	3	15	16							
SUBD	F8	F6	F2	4	7	8							
DIVD	F10	F0	F6	5	56	57							
ADDD	F6	F8	F2	6	10	11							
Estações de reserva					S1	S2	RS para j	RS para k	A				
	Tempo	Nome	Busy	Op	Vj	Vk	Qj	Qk					
	0	Add1	Não										
	0	Add2	Não										
	0	Add3	Não										
	0	Mult1	Não										
	0	Mult2	Não										
	0	Load1	Não										
	0	Load2	Não										
	0	Load3	Não										
Status dos registradores													
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30	
57		Qi		0	0		0	0	0				

# Exemplo com Valores

- Refazendo o exemplo anterior para os seguintes valores:
  - $\text{Mem}[34+R2] = 2;$
  - $\text{Mem}[45+R3] = 5;$
  - $F4 = 4;$



# Exemplo com valores

- Resolução no quadro





# Algoritmo de Tomasulo

- Vantagens:

- O algoritmo de tomasulo facilita a geração de código do compilador;
  - É possível alcançar alta performance sem que o compilador gere um código específico para o pipeline da arquitetura;
- Permite que o processador continue a executar instruções quando ocorre miss na cache;
  - Execução fora de ordem (Out-of-order - OoO);
- Reduz atrasos causados por hazards RAW;
- Eliminação de hazards WAW e WAR;

- Desvantagens:

- Complexidade do hardware;



# Exercícios

1. Execute o seguinte trecho de instruções e valor inicial para os registradores

ADD.D	F2, F4, F6	F4=2;
DIV.D	F8, F10, F2	F6=6;
SUB.D	F2, F6, F4	F10=36;
MUL.D	F10, F6, F4	

e descubra o ciclo final de execução de cada instrução, o estado das estações de reserva e o valor dos registradores em cada ciclo considerando:

- a. Um sistema com:
  - i. Uma unidade funcional capaz de realizar operações de soma e subtração;
  - ii. Uma unidade funcional capaz de realizar operações de divisão e multiplicação;
  - iii. 5 ciclos para soma e subtração, 10 ciclos para multiplicação e 20 ciclos para divisão
  - iv. Duas unidades de reserva para soma e subtração;
  - v. Duas unidades de reserva para multiplicação e divisão;
- b. Um sistema com as mesmas especificações do anterior, exceto nos itens i e ii, onde se tem duas unidades funcionais para cada “dupla” de operação;

# Referências

PATTERSON, D. A.; HENNESSY, J. L. Computer Architecture: A Quantitative Approach. Fifth Edition

