



Universidade de Brasília

Departamento de Ciência da Computação

Disciplina: CIC 116394 – Organização e Arquitetura de Computadores – Turma A
Prof. Marcus Vinicius Lamar

2012/1

Nome:

GABARITO

d0 d1 / d2 d3 d4 d5 d6 d7 d8
Matrícula: /

Prova 2

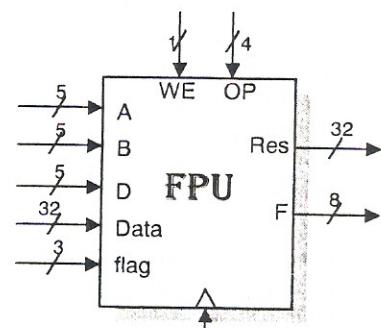
1)(6.0) No princípio da família x86, a FPU (*Float Point Unit*) evoluiu como um chip separado do processador principal desde o 8087, passando pelo 80187, 80287, 80387 até o 80487, que na verdade era um 486DX completo (com FPU incorporada) por jogada de marketing da Intel. Hoje em dia a Unidade de Ponto Flutuante é uma parte do processador que apenas não é implementada em chips de microcontroladores de muito baixo custo (brinquedos, etc), ou onde a aplicação específica não requeira esse tipo de operação matemática (MP3 player, etc). A maioria dos modernos processadores trabalha com a representação de números seguindo a recomendação IEEE 754 e realiza operações aritméticas em ponto flutuante com o auxílio de uma FPU incorporada ao processador principal.

Dada a implementação do Coprocessador 1 em IEEE 754 precisão simples do MIPS definida abaixo:

Instruções acrescentadas	
FS.FS	FD
FS.FX	PO
FS.FP	PD
FS.FD	FD
FS.FF	FF
FS.FF	PP
FS.FD	PD
FS.FF	FD
[20:18]	F
WT, (PS)	{
WT, (PS)	}
FD < FS	
WT < WT	
WT < 1	
OP	Operação
0000	Res=R[A]
0001	Res=R[A] + R[B]
0010	Res=R[A] - R[B]
0011	Res=R[A] x R[B]
0100	Res=R[A] / R[B]
0101	Res=sqrt(R[A])
0110	Res=abs(R[A])
0111	Res=neg(R[A])
1000	Res=int(R[A])
1001	Res=float(R[A])
1010	Res=X, F[flag] = R[A]=R[B]?
1011	Res=X, F[flag] = R[A]<=R[B]?
1100	Res=X, F[flag] = R[A]<R[B]?

OP	Operação
0000	Res=R[A]
0001	Res=R[A] + R[B]
0010	Res=R[A] - R[B]
0011	Res=R[A] x R[B]
0100	Res=R[A] / R[B]
0101	Res=sqrt(R[A])
0110	Res=abs(R[A])
0111	Res=neg(R[A])
1000	Res=int(R[A])
1001	Res=float(R[A])
1010	Res=X, F[flag] = R[A]=R[B]?
1011	Res=X, F[flag] = R[A]<=R[B]?
1100	Res=X, F[flag] = R[A]<R[B]?

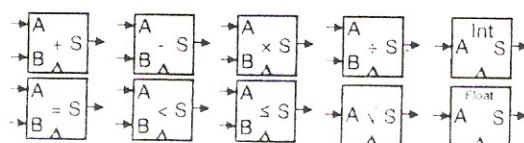
13 1101 Res=R[B]



Esta FPU efetua a multiplicação interna do clock de entrada usando um PLL, de forma que todas as operações são realizadas em menos de um único ciclo do clock externo. Em um ciclo de clock são efetuadas as seguintes tarefas:

- I) Escreve sincronamente (borda de subida) R[D]=Data, se WE=1
- II) Lê assincronamente R[A] e R[B] do Banco de Registradores interno
- III) Calcula assincronamente Res = R[A] op R[B] e atualiza F[flag]

- a) (3.0) Modifique o caminho de dados na folha em anexo e o controle do processador Uniciclo de modo que a definição de um pino externo FPUen=1, faça com que o processador reconheça as instruções definidas pela FPU e as processe corretamente, e caso FPUen=0, o Coprocessador 1 é ignorado pelo processador MIPS.
- b) (3.0) Apresente uma possível implementação do Coprocessador 1, mostrando seus elementos internos, interligações, multiplexadores e controle (caso seja necessário). Considere um banco de registradores similar ao do processador principal e os blocos funcionais dados abaixo.

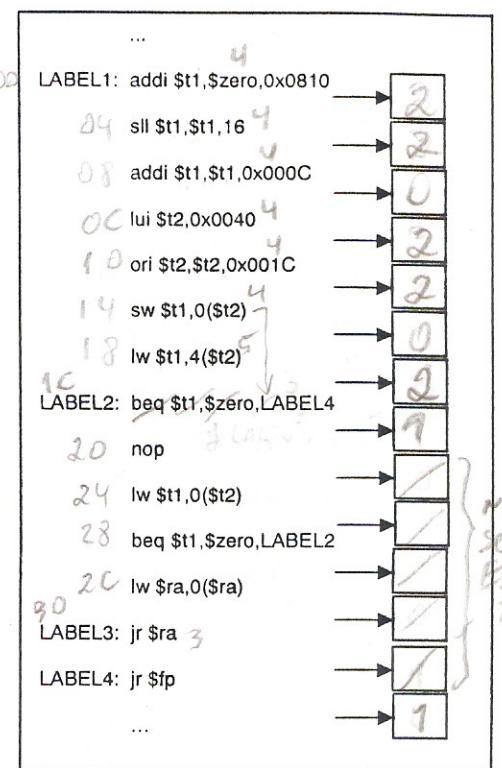


2) (4.0) Considerando apenas os seguintes tempos de atraso das unidades operativas do caminho de dados de uma CPU MIPS:

Unidade	Tempo
Operações lógicas (and, or, xor) com a ULA	80ps
Operações aritméticas simples (+, -) com a ULA	150ps
Operações aritméticas complexas (\times, \div) com a ULA	500ps
Leitura/Escrita no Banco de Registradores	50ps
Leitura da memória de Instruções ou Dados	300ps
Escrita na memória de Dados	150ps

Com relação ao trecho de programa em assembly MIPS ao lado, onde o LABEL1 corresponde ao endereço 0x00400000, e está sendo executado em *modo kernel*:

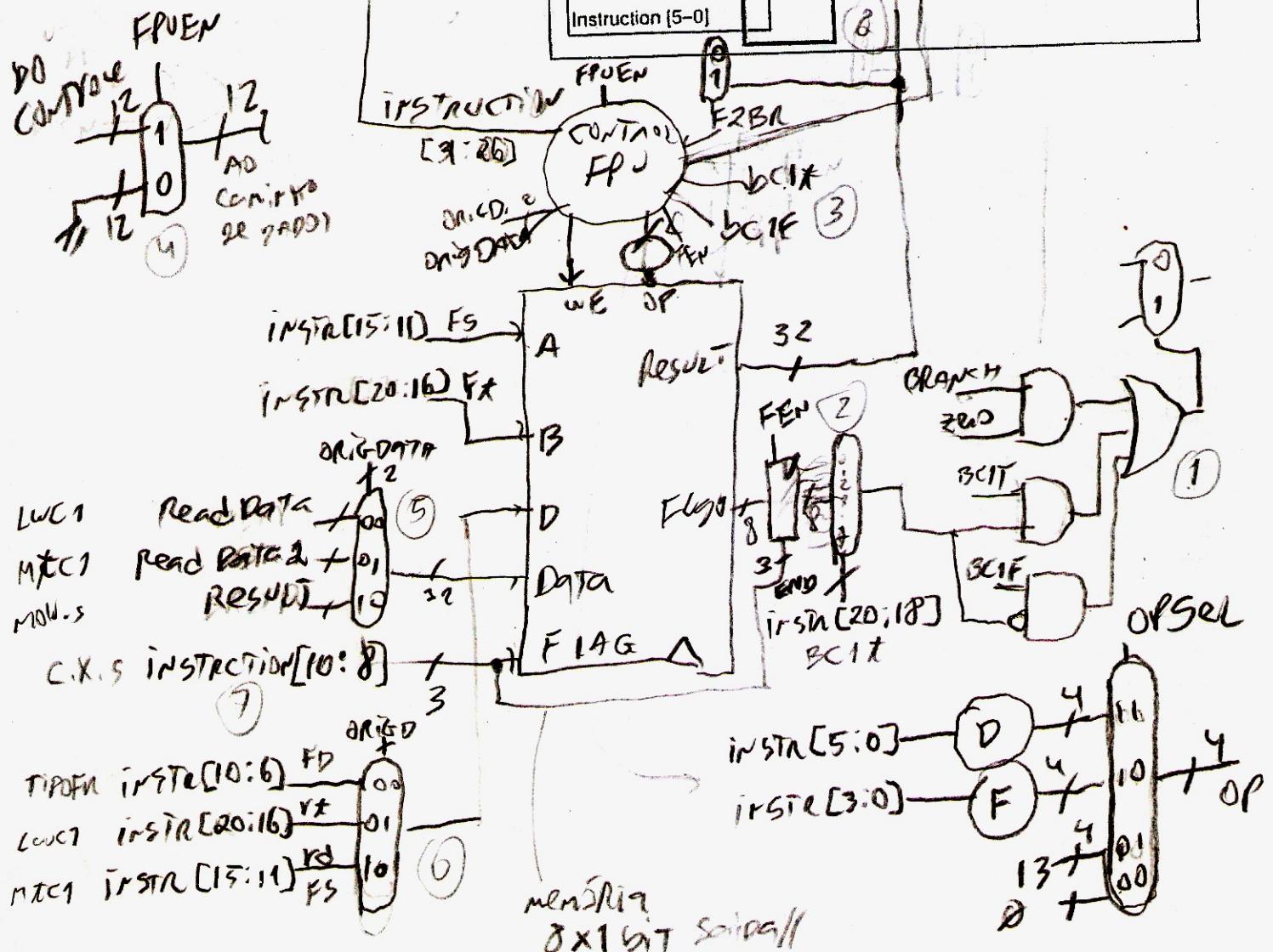
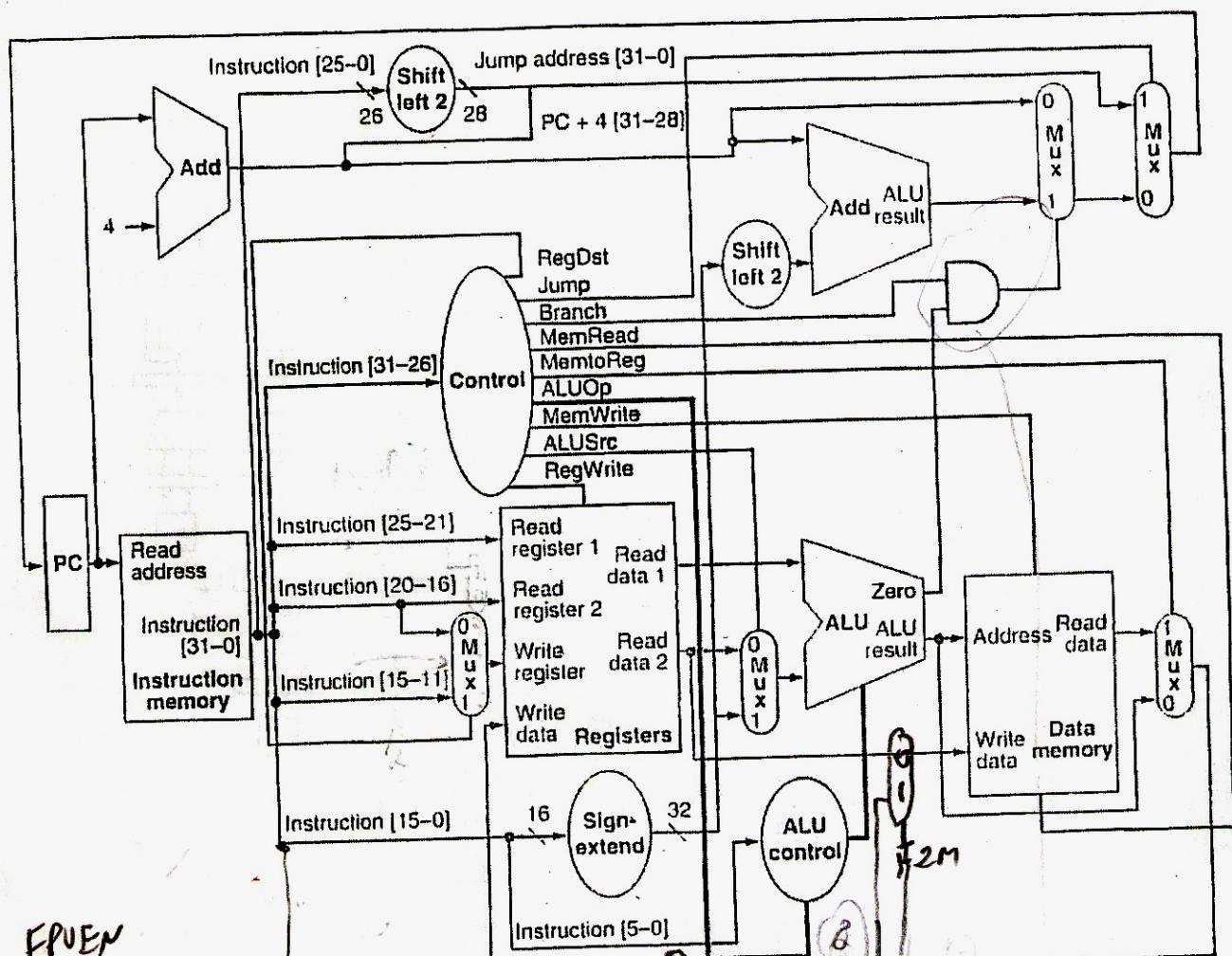
- a) (1.0) Para a implementação uniciclo, qual será a maior frequência de clock utilizável? Qual o tempo de execução deste trecho de código neste caso?
- b) (1.0) Para a implementação multiciclo vista em aula, qual será a maior frequência de clock utilizável? Qual o tempo de execução deste trecho de código neste caso?
- c) (1.0) Qual o tempo de execução da implementação pipeline, se todos os hazards forem tratados apenas com inserção de bolhas? Indique no espaço reservado o número de bolhas necessário (obs.: se não houver necessidade de bolha coloque 'zero'). *Bolhas 22 Escalver → 16*
- d) (1.0) Qual o tempo de execução para implementação em pipeline, se os hazards forem tratados eficientemente pelo processador com forwarding e/ou inserção de bolhas? (sem execução fora de ordem!) Preencha o pipeline esquemático na folha em anexo indicando as instruções, bolhas e forwards sugeridos. Considere que os registradores podem ser escritos e lidos no mesmo ciclo, que o branch não previsto é avaliado na etapa ID, e as instruções j, jal e jr necessitam 2 ciclos.



3)(1.0) E atendendo a pedidos da turma, responda V ou F:

- (✓) (0.2) A organização uniciclo é sempre mais rápida que a organização multiciclo para uma mesma frequência de clock.
- (✗) (0.2) Para um workload composto 2% de instruções add, 2% de mult, 1% de div, 15% de lw, 5% de sw, 40% de beq e bne, 35% de j, jal e jr, a organização multiciclo sempre será mais rápida que a uniciclo caso sejam usadas as mesmas unidades funcionais.
- (✗) (0.2) A implementação em pipeline sempre é mais eficiente que a Uniciclo ou a Multiciclo, independentemente do workload. *DEPENDE DA FREQ! HAZARDS*
- (✗) (0.2) A ausência da ULA de ponto flutuante é um exemplo de hazard de estrutural.
- (✗) (0.2) Em programa que calcula a soma dos n termos de um vetor pode-se explorar somente a localidade espacial dos dados e a localidade temporal do código.

Boa Sorte!!!





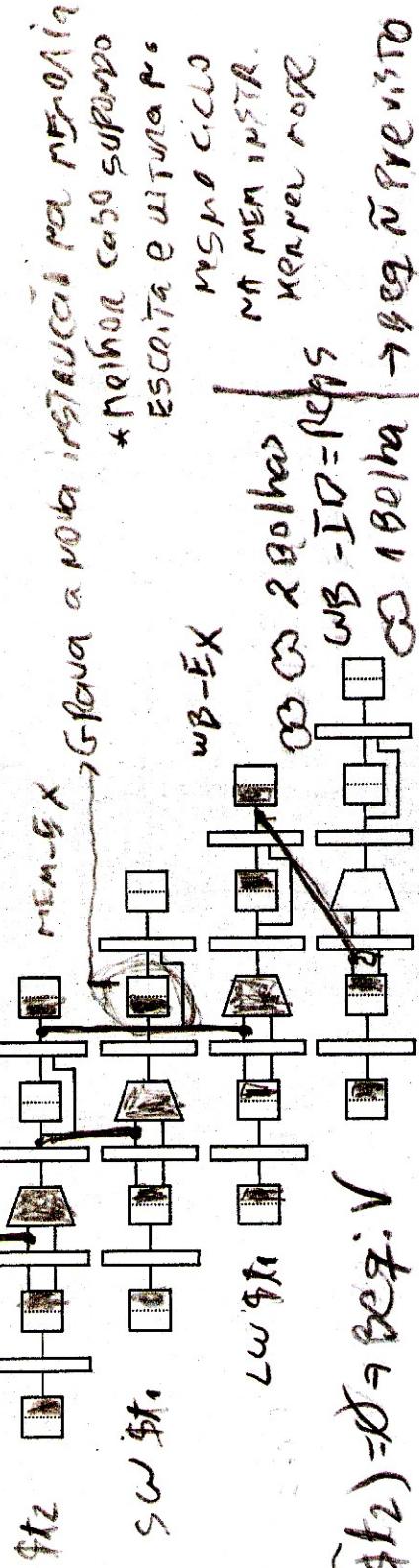
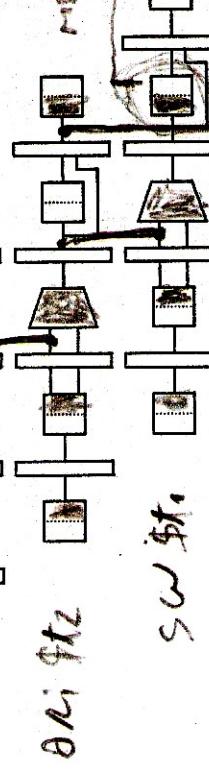
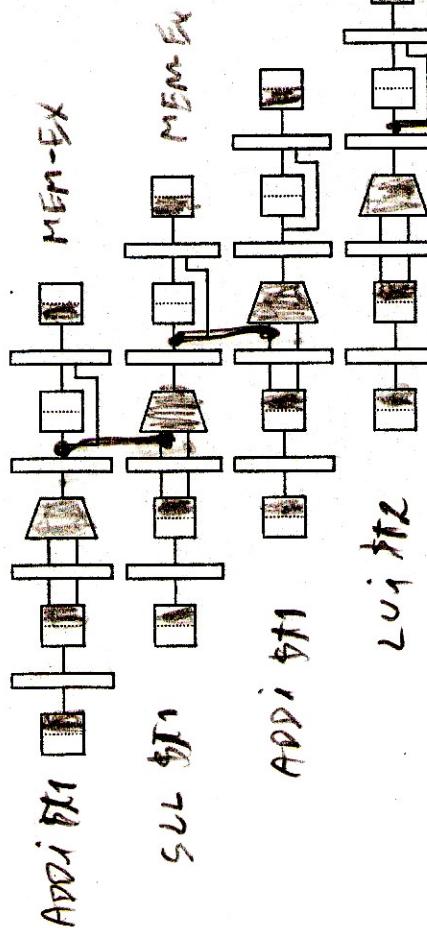
Universidade de Brasília

Departamento de Ciência da Computação

Disciplina: CIC 116394 – Organização e Arquitetura de Computadores
Prof. Marcus Vinicius Lamar

Nome:

Matrícula: _____



$$ADD = MEM(4 + \#T_2) \Rightarrow \text{Resg: } V$$

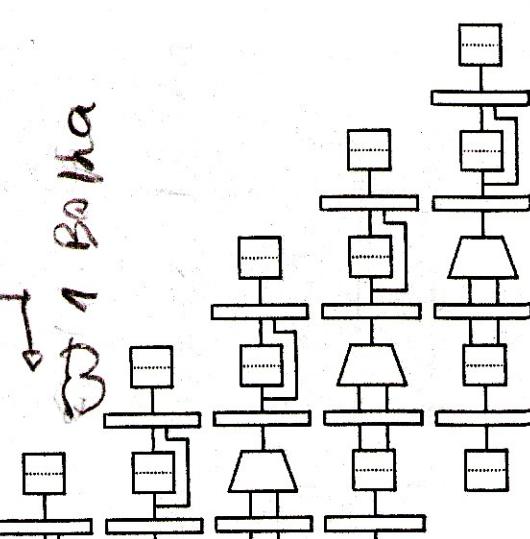
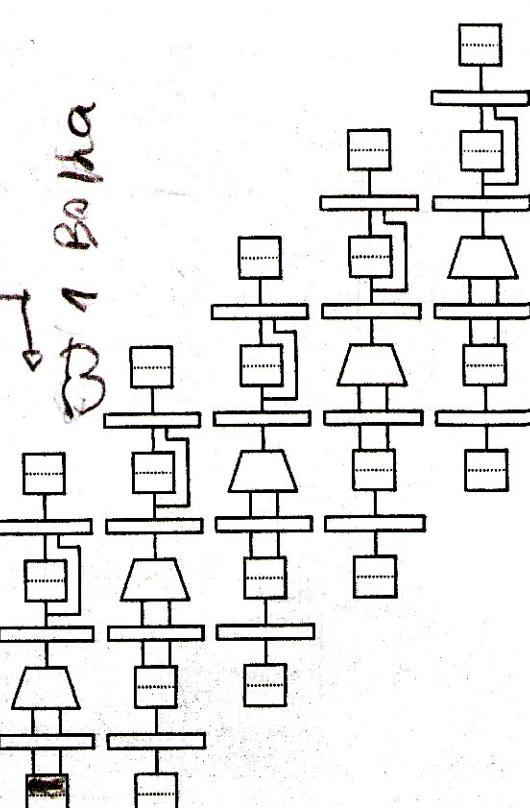
Beg Tomas LABEL 4º

JR \$fp

1 Bolha

neste ciclo:

A mem irá m(C+T2) é escrita
e A mem irá m(T1) é lida



2º PROVA GABARITO

1) Captura de mips unicycle normal
e contas síntesis de cintos da F10 em Zéro

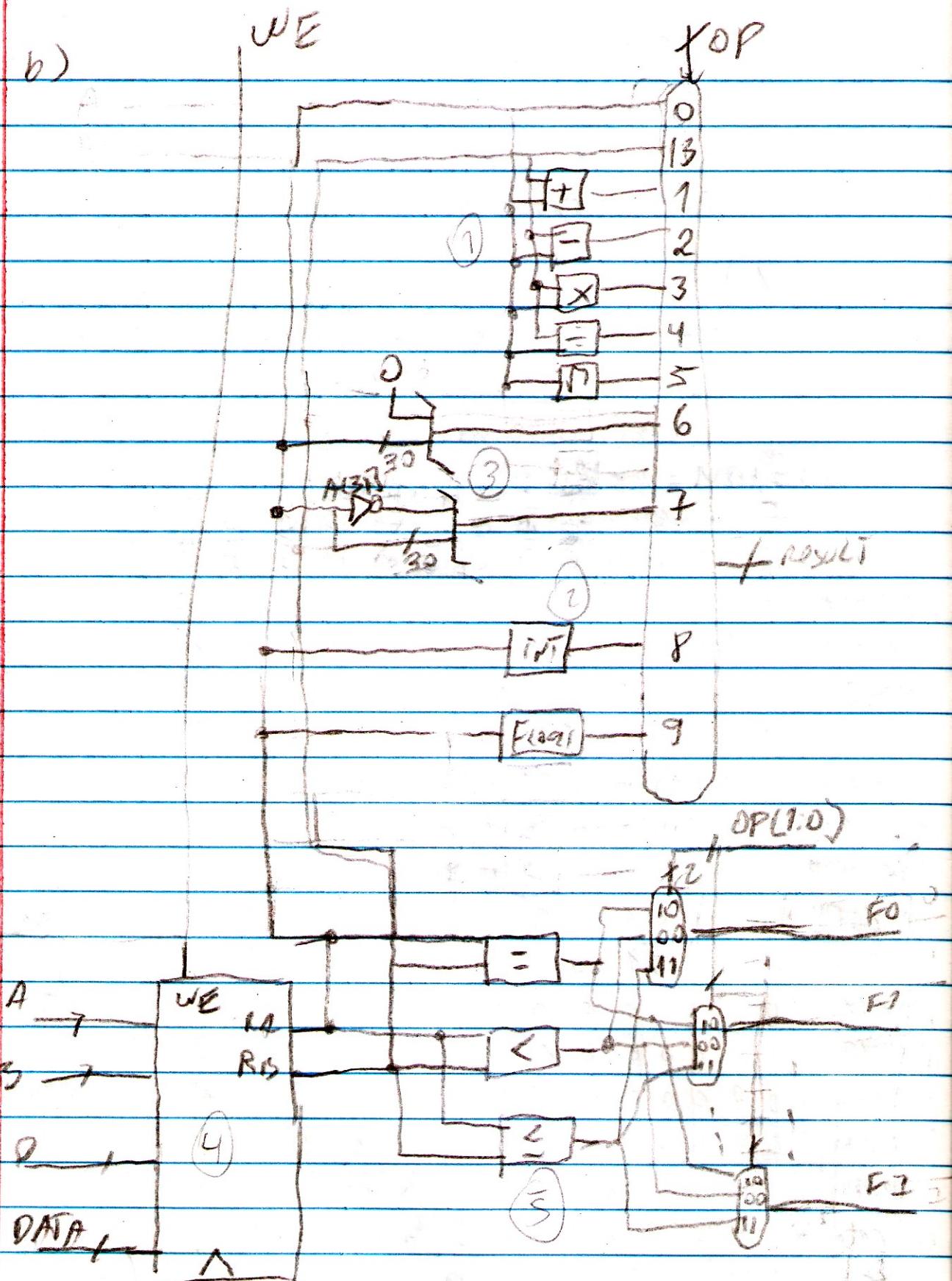
DP. (025) 8

Catmolo:

64A618905

	TIPDR	LW	SW	BEG	TIPEN	LWC1	SCL	BCDT	BCIF	C.X.S	MRS	NTC1	MFC1
REGDST	1	0	0	0	0	0	0	0	0	0	0	0	0
ORIGALU	0	1	1	0	0	0	1	1	0	0	0	0	0
MEM2REG	0	1	0	0	0	0	0	0	0	0	0	0	0
ESCRREG	1	1	1	0	0	0	0	0	0	0	0	1	0
LEMEM	0	1	0	0	0	1	0	0	0	0	0	0	0
ESCRMEM	0	1	0	0	0	0	1	0	0	0	0	0	0
BRANCH	0	0	0	1	0	0	0	0	0	0	0	0	0
ALUOP	1	0	0	0	0	1	0	0	0	0	0	0	0
J	0	0	0	1	0	0	0	0	0	0	0	0	0
BC1X	0	0	0	0	0	0	0	1	0	0	0	0	0
BC1F	0	0	0	0	0	0	0	0	1	0	0	0	0
F2BR	0	0	0	0	0	0	0	0	0	0	0	0	1
F2M	0	0	0	0	0	0	1	0	0	0	0	0	0
ORIGD	0	0	0	0	0	0	0	1	0	0	0	0	0
ORIGDATA	0	0	0	0	0	0	0	0	0	0	1	0	0
WE	0	0	0	0	1	1	0	0	0	1	1	0	0
OPSEL	0	0	0	0	0	0	1	3	0	0	F	0	0
FEN	0	0	0	0	0	0	0	0	1	0	0	0	0

b)



Registros de Fregas (6)
Aqui ou no DP

2) UNICICLO pen

$$a) \text{TIPO-R} \text{Ldg} = 300 + 50 + 80 + 50 = 480$$

$$\text{TIPO-R RMT} = 300 + 50 + 150 + 50 = 550$$

$$\text{TIPO-R Comp} = 300 + 50 + 500 + 50 = 900$$

$$LU = 300 + 50 + 150 + 300 + 50 = 850$$

$$SW = 300 + 50 + 150 + 150 = 650$$

$$BEG = 300 + 50 + 150 = 500$$

$$JR = 300 + 50 = 350$$

$$\log(0) f_{ori} = \frac{1}{900p} = 1,11 \text{ GHz} //$$

$$xt1 = 0x0810000c \rightarrow 04000030 \rightarrow \text{LABEL3}$$

$$xt2 = 0x0040001c$$

I CPI T

$$t_{ori} = 9 \times 1 \times 900p = 8,1 \text{ ns} //$$

b) MULTICICLO ENTRADA + SAIDA

$$f_{mult} = \frac{1}{500p} = 2 \text{ GHz} //$$

$$t_{pout} = (4+4+4+4+4+4+5+3+3) \times 500p$$

$$t_{fout} = 17,5 \text{ ns} //$$

c) Considerando

Beg Avaliada ~ 2º Ciclo

Banco Reg. Para Escrita e Ler no mesmo ciclo

$$t_{ope} = (9+12) \times 500p = 10,5 \text{ ns} //$$

$$d) t_{ope} = (9+4) 500p = 6,5 \text{ ns} //$$

Explicando melhor

Para Uniciclo e Multiciclo:

BEQ é sobreescrita com #LABEL3

Para Pipe Line:

A memória só será escrita no 4º ciclo
o que permite que o BEQ seja executado
antes de ser sobreescrito.

Como $st_1 = \text{MEM}(4 + st_2) = \text{NOP} = \text{ZENO}$
o BEQ não previsto é tomado
executando entre a instrução JR #f/p