UFRN - DCA

Circuitos digitais

Aluno: Teophilo Vitor de Carvalho Clemente.

20190080555

Segunda lista

Questões:

- Lista normal:
  1, 2, 6, 7, 9, 10, 11, 12, 13, 14.

- Lista complementar:
  5, 6 e 7

TOTAL = 13 questões

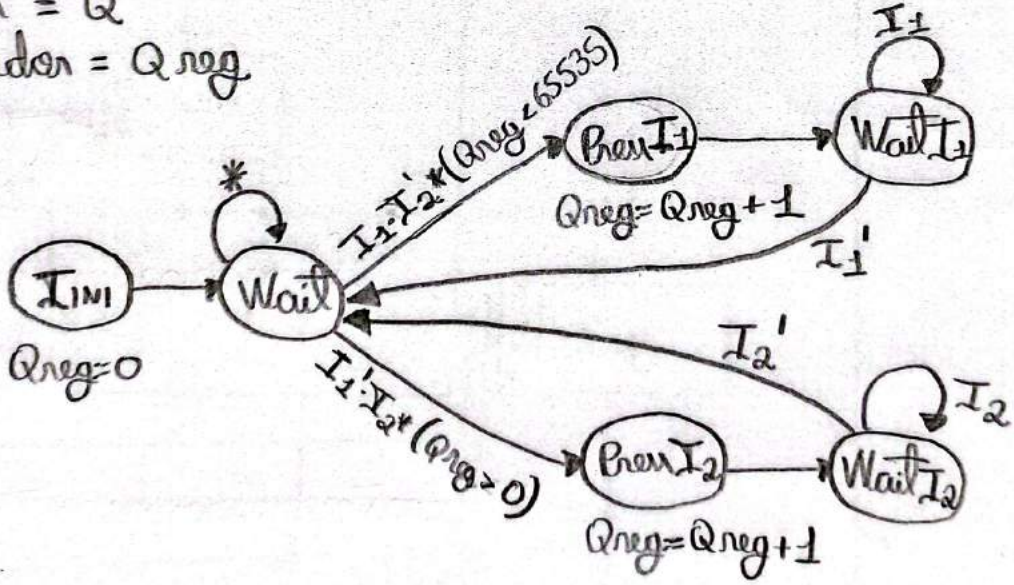
Lista normal

Q1

- Dados e comportamentos:
- Entradas = I_1 e I_2 com 1 bit cada.
- Saídas = Q, onde seu valor inicial é 0 e possui 16 bits.
- Se I_1 for pressionado, incrementamos Q.
- Se I_2 for pressionado, decrementamos Q.
- Caso ambos sejam pressionados, o sistema não fará nada.
- O sistema não reinicia a contagem, e não ultrapassará o maior valor de Q nem será menor que 0.
- Maior valor $\Rightarrow 2^{16} - 1 = 65535$.
- Com isso podemos criar a máquina de estados finitos de alto nível:

$$* = [I_1 \cdot I_2' * (Qreg < 65535) + I_1' \cdot I_2 * (Qreg > 0)]'$$

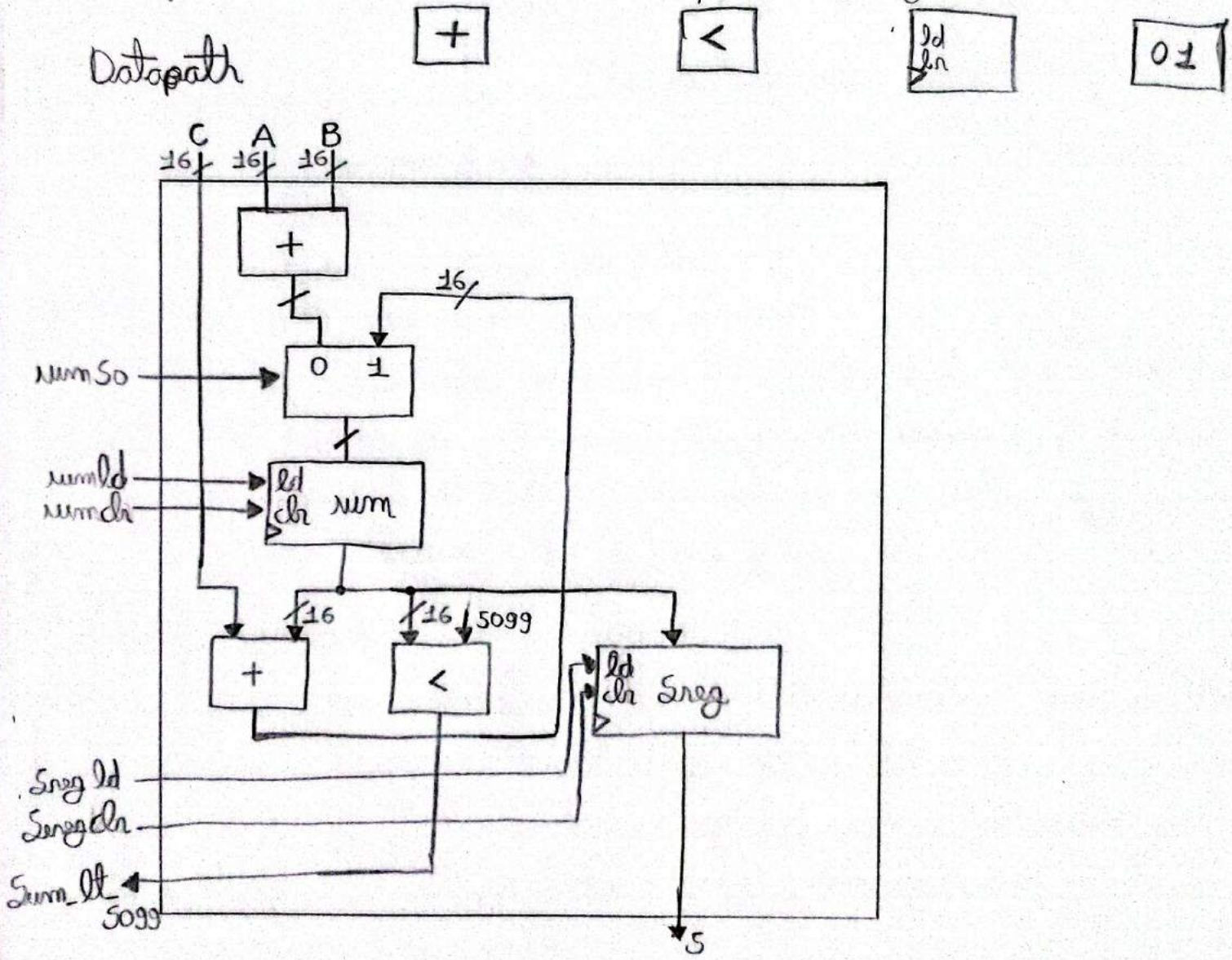
Entradas = I_1 e I_2
Saídas = Q
Registrador = Qreg



(Q2) Comportamento:

- Somar A + B e armazenar em num.
- Somar C ao valor de num e armazenar.
- Armazenar o valor de num em Sreg.
- Verificar se o valor do soma é menor que 5099.
- Componentes: Somador, Comparador, Registrador, Multiplexador

Datapath



(Q6) Determine o caminho crítico do Mux 4x1.

Atrasos → Inversor: 1 ns
Portas: 2 ns
Fios: 1 ns

O caminho crítico passará por um inversor, uma porta AND e uma porta OR, e teremos 5 segmentos de fios, o da entrada, o da ramificação até o inversor, o do inversor até a porta AND, o da porta AND até a OR e o fio da saída.

∴ O delay total será = 1 + 2 + 2 + 5 = 10 ns //

(Q7) Determine o caminho crítico do somador completo.

Atrasos → Inversor: 1 ns
Portas: 2 ns
Fios: 1 ns ou 0 ns

a) Com fios que não geram atraso:

O caminho crítico do somador completo é de 4 ns (2 ns da porta AND + 2 ns da porta OR). Tendo no somador de propagação carry de 8 bits, 8 somadores completos encadeados, o delay total será → 8 · 4 = 32 ns //

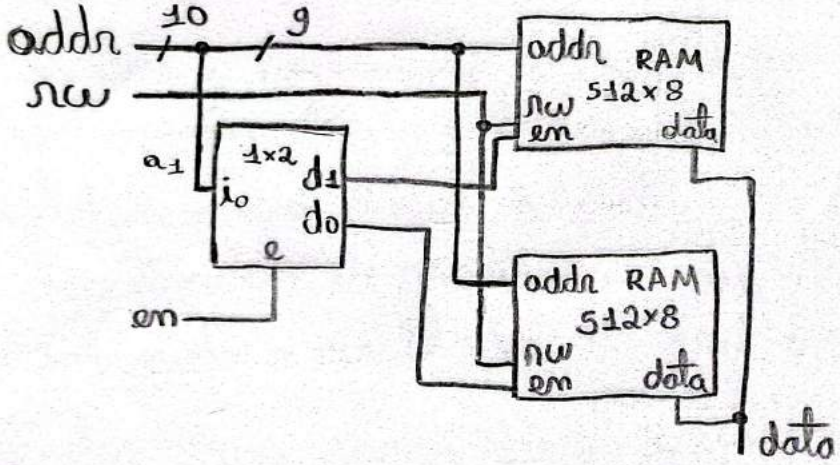
b) Com fios que geram atraso:

Cada somador completo possui 1 fio entre a porta AND e OR, assim 8 ns no total. Além disso, há o fio que conecta a saída de um somador completo com a entrada do outro, assim 7 fios no total (7 ns) e os fios C_i e C_o que juntos somam 2 ns, então:

Delay total = 32 ns + 8 ns + 7 ns + 2 ns = 49 ns //

(Q9) Projete uma memória RAM de 1024×8 usando apenas RAMs 512×8.

- Para isso usamos —— ligar RAMs de forma que se somem.
- Para isso, precisamos de um decodificador e 10 linhas, 9 serão utilizadas no deslocamento da memória e 1 para o decodificador habilitar em qual memória será guardada os dados
- Como temos uma memória RAM ela pode ser escrita e lida, precisando da entrada rw.



(Q10) Tendo que a lista de instruções de um processador é uma sequência binária de bits, o número máximo de palavras é 2^n, onde n é a largura de bits do processador. Sendo assim, como n=20, temos $2^{20} = 1.048.576$ palavras.

Q 11

a) Copiar a informação de uma posição na memória para outra posição na memória.

Falsa, é possível apenas com dois ciclos de clock, pois com a operação load carrega-se os dados da memória para o registrador de arquivos e a operação store armazena os dados do registrador de arquivos em uma nova posição.

b) Copiar dois endereços do registrador em duas posições de memória.

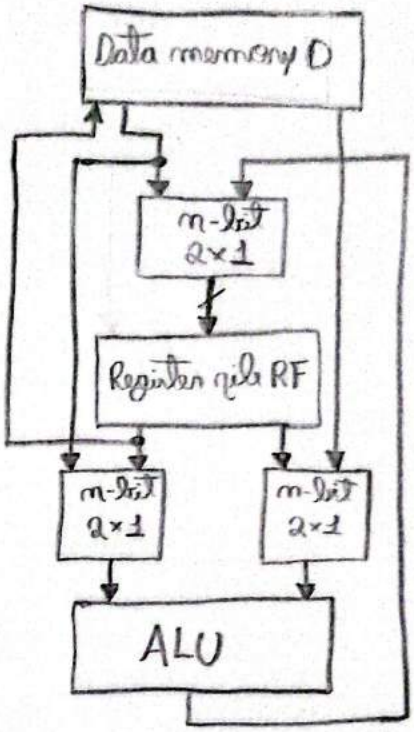
Falsa, pois apenas uma cópia de endereço do registrador é permitido por ciclo, ou seja, para isso precisaríamos de dois ciclos de clock.

c) Adicionar dados a partir do endereço de um registrador de arquivo e uma posição de memória, salvando o resultado na mesma posição de memória.

Falsa, para fazer essas operações são necessário 3 ciclos de clock; um para carregar os dados (load), um para adicionar os dados (add) e um para realizar o armazenamento dos dados (store).

(Q12)

Para realizar a modificação vamos precisar de dois multiplexadores 2×1. No caso, duas posições da memória são lidas do componente Data memory D e, através das entradas dos multiplexadores na ALU, elas são alimentadas na ALU. A seguir, o resultado da ALU irá passar para o MUX de entrada do registrador de arquivos e armazenar o endereço no local apropriado.



(Q13) D[8] = (D[4] + D[5]) - D[7]

1) Operação load: carregar o dado D[4] para o registrador de arquivos R[0].
2) Operação load: carregar o dado D[5] para o registrador de arquivos R[1].
3) Operação ALU: adicionar R[0] e R[1] e armazenar o resultado no registrador de arquivos R[2].
4) Operação load: carregar o dado D[7] para o registrador de arquivos R[0].
5) Operação ALU: subtrair R[0] e R[2] e armazenar o resultado no registrador de arquivos R[1].
6) Operação store: armazenar o valor de R[1] na posição D[8] da memória de dados.

(Q14) Código assembly para D[4] = D[1] * 2 + D[2]

Comandos: MOV Ra, d → load
MOV d, Ra → store
ADD Ra, Rb, Rc → ALU add

Código:

```
MOV R0, 1   // load da memória D[1] para o registrador RF[0]
ADD R0, R0, R0  // R[0] = R[0] + R[0]
MOV R1, 2   // load RF[1] = D[2]
ADD R0, R0, R1 // R[0] = R[0] + R[1]
MOV 4, R0   // store D[4] = RF[0]
```

Lista complementar

(Q5) Diferenciar entre SRAM e DRAM.

Memórias SRAM precisam de seis transistores por célula, por outro lado as memórias DRAM utilizam apenas um transistor e um capacitor. Devido a isso, a memória é mais cara e maior que uma DRAM que pode armazenar o mesmo número de bits. Por outro lado, SRAMs, em geral, apresentam o tempo acesso mais rápido em relação às DRAMs, devido que esta última precisa realizar uma atualização periódica (refresh) do seu conteúdo por causa da descarga do capacitor ao longo do tempo, bloqueando assim o acesso ao conteúdo durante esse refresh.

(Q7) Quantas instruções (opcode) podem ser suportadas pelo processador de 4 bits.

Uma instrução opcode é uma sequência binária de bits que indica a operação da instrução que será realizada. Desse modo, temos 2^n possibilidades, onde n é a largura de bits do processador, como o processador em questão é de 4 bits então:

$\Rightarrow 2^4 = 16$ instruções.

(Q6) Projetar uma ROM de 1024×8 com ROMs 512×4

Passos:

- Ligar duas ROMs de forma que os bits de dados fiquem em paralelo.
- Ligar duas ROMs de modo que elas se somem.

Assim, temos que ter um total de 4 ROMs com 10 linhas de endereço, sendo 9 delas para representar o deslocamento na ROM e 1 linha (aq) que irá para o decodificador habilitar o bloco superior ou inferior de ROMs. Além disso, como essa memória é apenas de leitura, ela não precisa da entrada "rw", ficando com "en" e "addr" como entradas e "data" como saída.