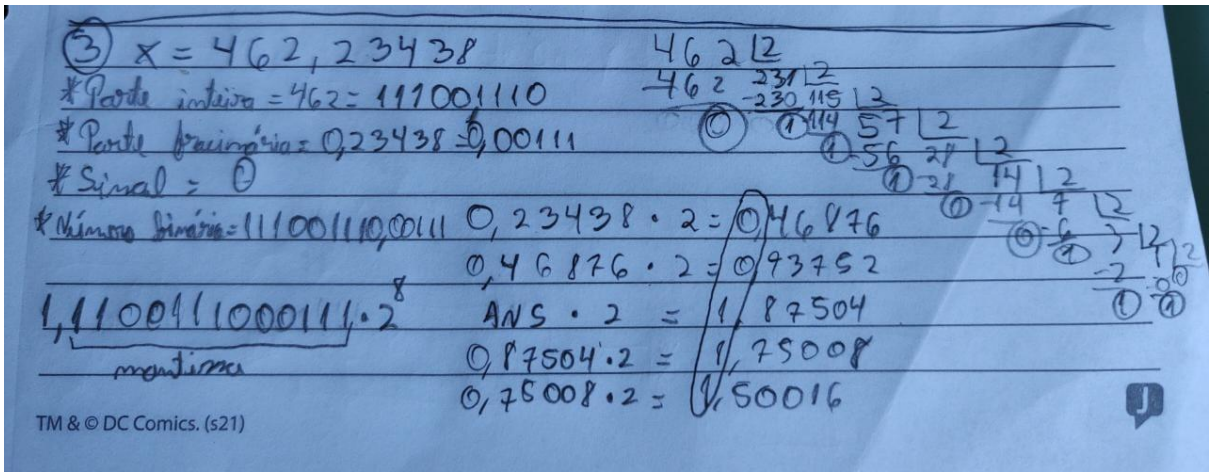
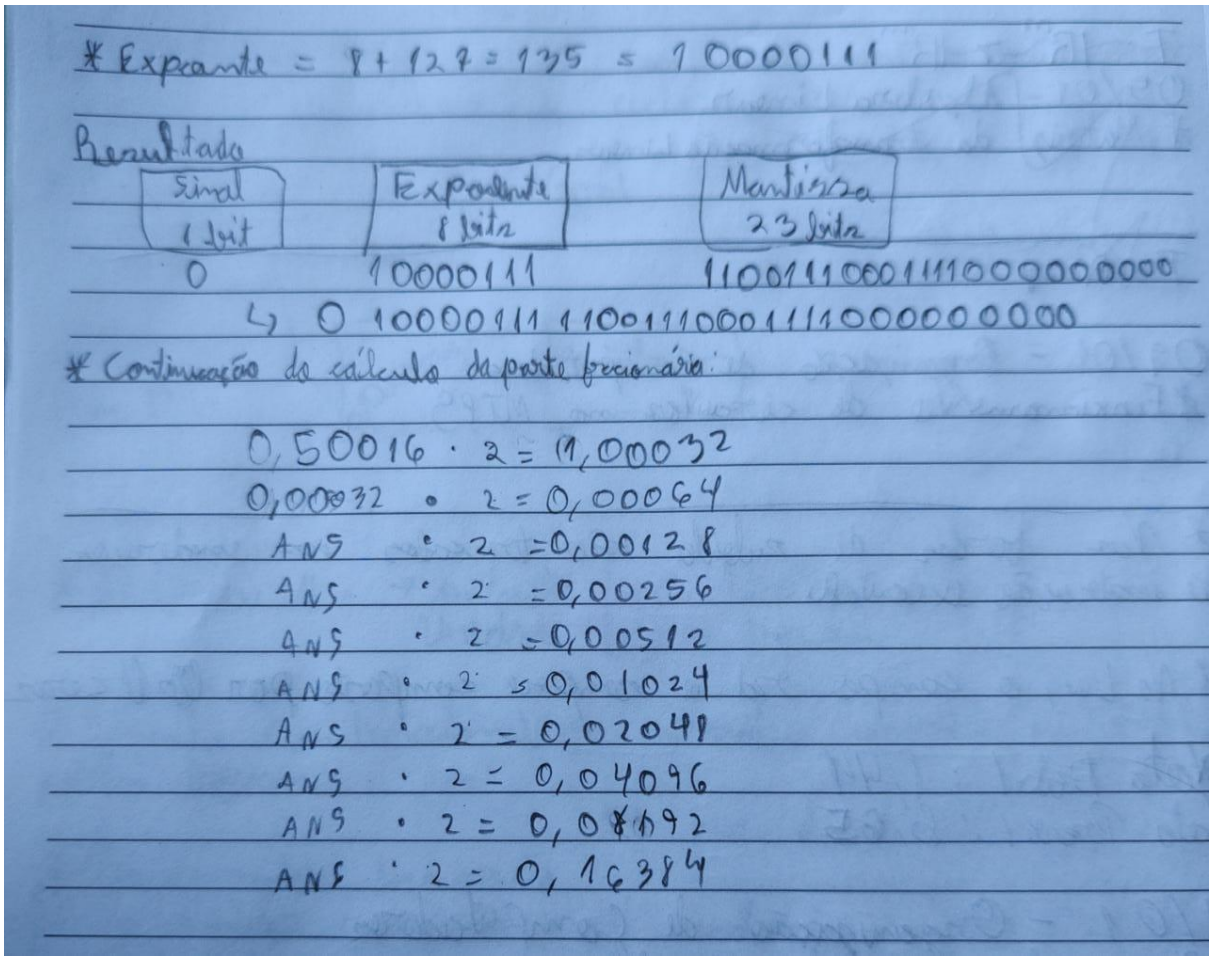
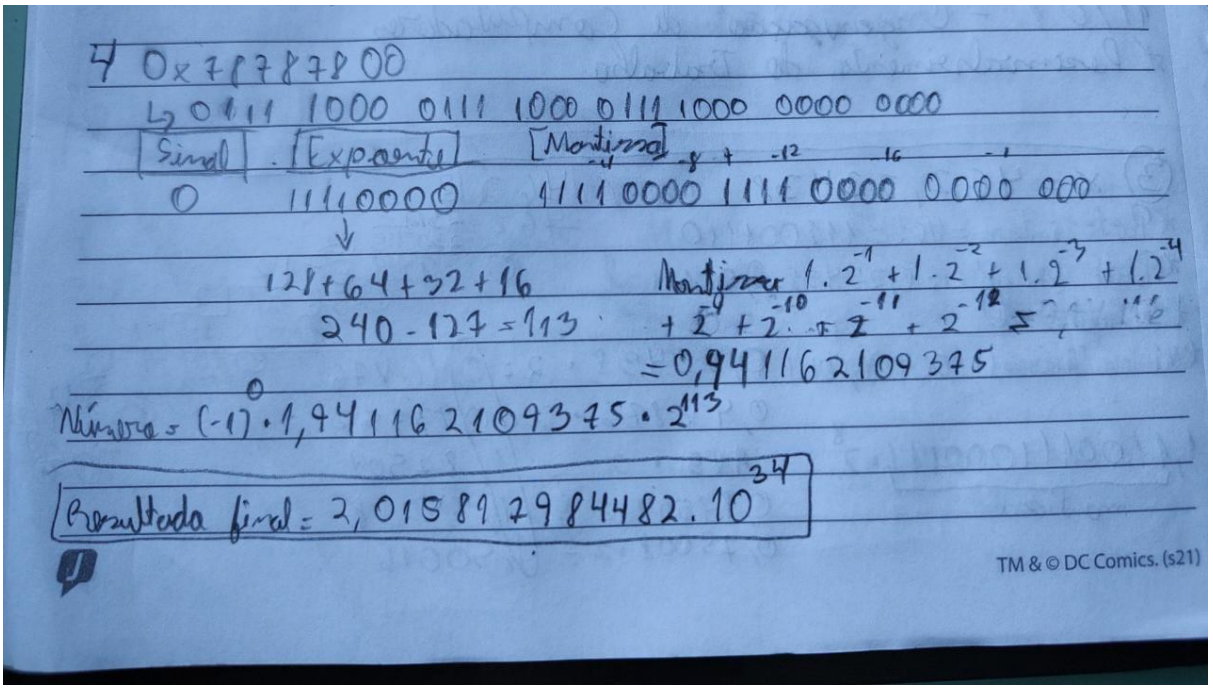
Questão 3)

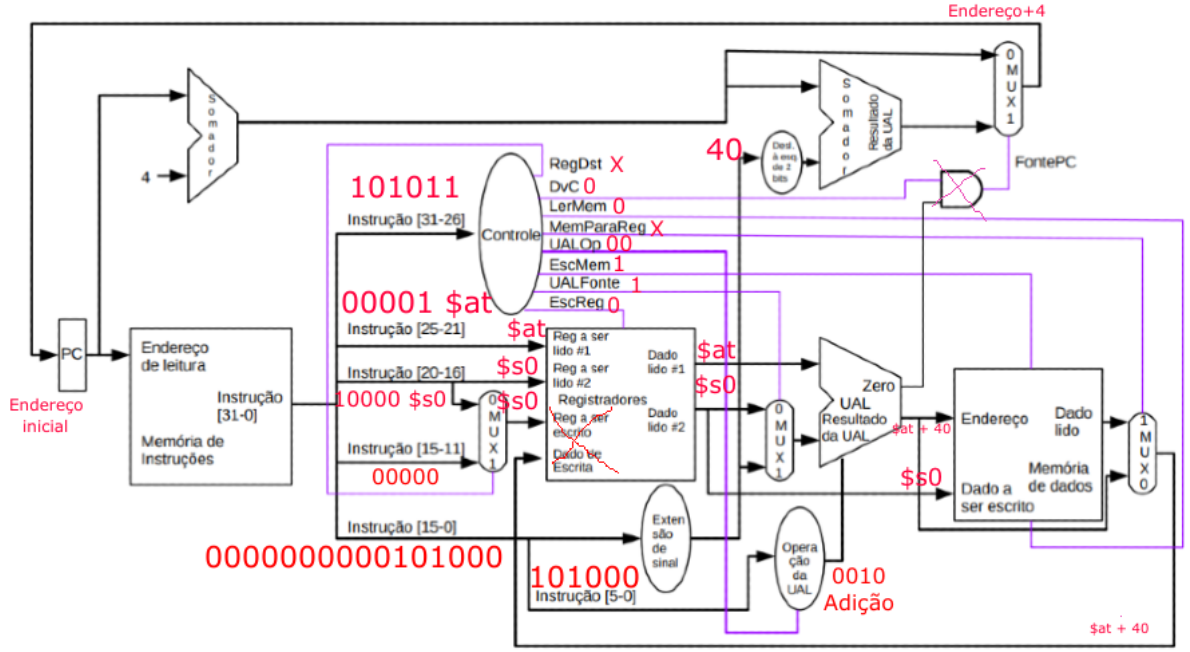

③ x = 462,23438
* Parte inteira = 462 = 111001110
* Parte fracionária = 0,23438 = 0,00111
* Sinal = 0
* Número binário = 111001110,00111
1,11001110001111 · 2^8
mantissa
0,23438 · 2 = 0,46876
0,46876 · 2 = 0,93752
ANS · 2 = 1,87504
0,87504 · 2 = 1,75008
0,75008 · 2 = 1,50016
462 | 2
462 231 | 2
-230 115 | 2
1 114 57 | 2
1 56 28 | 2
1 28 14 | 2
0 14 7 | 2
0 6 3 | 2
1 2 1
0 1
TM & © DC Comics. (s21)


* Expoente = 8 + 127 = 135 = 10000111
Resultado
Sinal
1 bit
Expoente
8 bits
Mantissa
23 bits
0
10000111
11001110001111000000000
↳ 0 10000111 11001110001111000000000
* Continuação do cálculo da parte fracionária:
0,50016 · 2 = 1,00032
0,00032 · 2 = 0,00064
ANS · 2 = 0,00128
ANS · 2 = 0,00256
ANS · 2 = 0,00512
ANS · 2 = 0,01024
ANS · 2 = 0,02048
ANS · 2 = 0,04096
ANS · 2 = 0,08192
ANS · 2 = 0,16384

Questão 4)



Questão 5) sw $s0, 40($at)

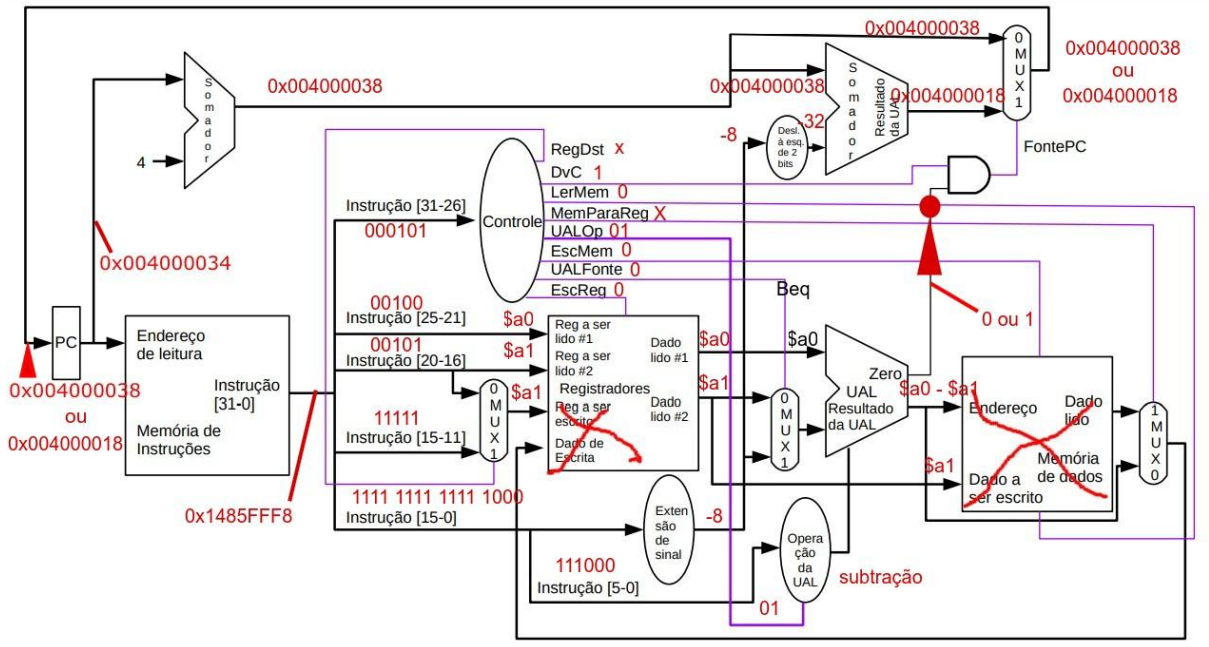


sw é uma instrução tipo I, convertendo a instrução para binário temos:

opcode (31-26)	rs (25-21)	rt (20-16)	immediate (15-0)
101011	00001	10000	0000000000101000

1) PC armazena o endereço das instruções, recebe o endereço respectivo a instrução sw $s0, 40($at) e envia o endereço para a memória de instruções.
2) A instrução em código binário é dividida em 5 partes e mandadas para partes diferentes do circuito.
3) [31-26] O opcode é enviado para o controle e suas saídas estão representadas na imagem, que foi baseada na tabela 1.
4) [25-21] O registrador rs, no caso o $at é enviado para o reg a ser lido #1
5) [20-16] O registrador rt, no caso o $s0 é enviado para o reg a ser lido #2 e enviado para o multiplexador.
6) [15-11] Parte do immediate, onde normalmente estaria o campo rd, mas como essa é uma instrução tipo I não R esta parte é irrelevante.
7) [15-0] O immediate é enviado para o circuito que faz a extensão de sinal e parte dele onde ficaria o campo funct [5-0] é enviado para a operação da unidade lógica e aritmética que tem como saída 0010 que representa adição.
8) O código da adição é enviado para a UAL que faz a operação recebendo o registrador $at e, já que UAL fonte é igual a 1, recebe o immediate 40.
9) A unidade lógica e aritmética entrega a memória de dados o endereço representado por $at + 40 e o dado a ser escrito representado por $s0, cuja saída entra no multiplexador junto com $at + 40, como MemParaReg é X não é relevante qual dos dois é escolhido.
10) Ao fim do processamento da instrução o circuito soma o endereço da instrução atual com 4 e segue para a próxima instrução

Questão 6) bne $a0, $a1, loop

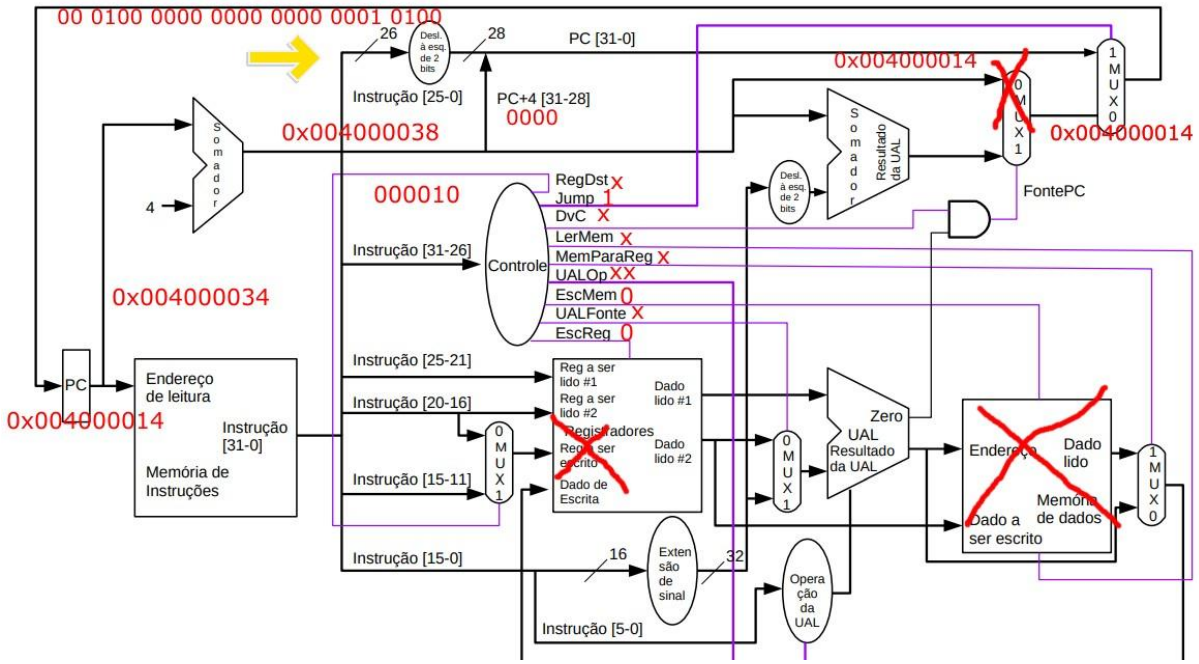


A instrução bne é de tipo I, convertendo a instrução para binário fica:
0x1485FFF8

opcode (31-26)	rs (25-21)	rt (20-16)	imm (15-0)
0001 01	00 100	0 0101	1111 1111 1111 1000

1) PC armazena o valor da instrução atual o endereço da instrução atual e manda para a memória de instruções que repassa a instrução 0x1485FFF8, para os demais circuitos.
2) O opcode 000101 é enviado para o controle que produz as saídas mostradas na imagem e extraídas da tabela 1.
3) O registrador $a0 00100 é enviado para Reg a ser lido #1 e $a1 00101 é enviado para Reg a ser lido #2.
4) Tanto o registrador $a1 quanto os bits [15-11] entram no multiplexador cujo controle é o RegDst, no entanto essa parte do circuito não é relevante.
5) O immediate [15-0] é enviado para a extensão de sinal que tem como saída o valor -8.
6) Esse valor em seguida é enviado para o circuito que faz o deslocamento dos bits 2 casas para a esquerda e o transforma em -32 que entra no somador com o endereço 0x004000038 e tem como saída 0x004000018.
7) O endereço 0x004000038 é obtido através da soma do endereço inicial extraído de pc com 4 pelo somador no topo superior esquerdo da figura.
8) Esse endereço então, como descrito anteriormente, entra no somador, mas também entra no multiplexador controlado pela porta not zero extraída da ual e pelo sinal de controle DvC.
9) Se o resultado $a0 - $a1 gera 0 o sinal zero será ativado e negado em seguida e enviado para porta and junto com DvC, portanto fonte pc será 0 e não ocorrerá desvio.
10) Se o resultado $a0 - $a1 gerar algo diferente de 0 o sinal zero não será ativado e será negado e enviado para porta and junto com DvC, portanto fonte pc será 1 e ocorrerá o desvio.

Questão 7) j loop



jump é uma instrução tipo I, convertendo para binário temos:

opcode(31-26)	endereço(25-0)
000010	00 0100 0000 0000 0000 0001 0100

1) PC armazena o endereço das instruções que recebe, no caso a instrução j loop, que é 0x004000034, e envia o endereço para a memória de instruções.
2) Em seguida o endereço da instrução vai até o somador no qual é somado 4 e se torna 0x004000038 que não será usado já que esta é uma instrução jump.
3) A instrução em código binário é dividida em 5 partes e mandadas para partes diferentes do circuito, no entanto como é uma instrução jump a única parte que é relevante é o opcode [31-26] que será mandado para o controle.
4) As saídas do controle estão representadas na imagem, que foi baseada na tabela 1.
5) O endereço [25-0] é enviado para a parte superior do circuito onde será realizado seu deslocamento em dois bits a esquerda.
6) Em seguida o endereço é enviado para o multiplexador controlado pela chave de controle Jump, que como está ligada como 1 manda o endereço que deve o programa deve pular para PC.
7) Por fim PC recebe o endereço para pular para e o programa continua a partir dali.