Lab 6 – Practice for Computer Arithmetic

Monday, March 22, 2021 11:20 AM

1. Consider Fig. B.5.12 (from book/lecture note) and explain clearly how ***slt*** instruction is manipulated in the 32-bit ALU.
   Also, explain why control signal values for ***slt*** instruction are '0111'.

The slt instruction is done by performing A-B and taking the resulting sign bit as the result of the operation. This works because if A < B then A-B will be negative and result in a sign bit of 1 showing that A is less than B. If A >= B then the resulting sign bit will be 0 showing that A is not less than B.

The control signal for the slt instruction is 0111 because the control signal is just a combination of bits that are input in the ALU. The control signal is representative of the bits A inv, B inv, S0, and S1 in this order. S0 and S1 represent the bits which control the mux that selects the output to choose. In this case A inv is set to 0 and B inv is set to 1 to allow for A-B to be performed. The sign bit from the resulting operation can then be chosen as the result by setting S0 and S1 to 1 which is the signal to select the slt function as the result. This is why the corresponding control signal for slt is 0111.

2. Consider the unsigned number multiplication hardware that we discussed in class.
   Assume that the hardware is used for unsigned 4-bit number multiplication [0101 * 1110].
   Show the contents of registers for each step of the multiplication using the 2-step multiplication algorithm.

MD	AC	MQ
….	….	….

0101 * 1110 = 0010 1010   5 * 14 = 70

MD	AC	MQ	Step	Iterations
0101	0000	1110		0
	0000	1110	AC+0	1
	0000	0111	>>	1
	0101	0111	AC+MD	2
	0010	1011	>>	2
	0111	1011	AC+MD	3
	0011	1101	>>	3
	1000	1101	AC+MD	4
	0100	0110	>>	4

Final result is 0100 0110 which is 70 in decimal, 5 * 14 = 70

3. Consider the Booth's algorithm for 2's complement number multiplication.
   (a) Show Booth's recoded number for binary value [1010011101] and justify your answer.
   (b) For [1010 * 0101], i.e., (-6) * 5 in decimal, show the contents of registers for each step.

MD	AC	MQ	MQ_{-1}
….	….	….	

A) Booth's recoded number for 1010011101 is ^1 1 ^1 0 1 0 0 ^1 1 ^1

This can be shown by creating a table containing the values and converting them to their equivalent in decimal

Place	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Booth's values	^1	1	^1	0	1	0	0	^1	1	^1
Decimal	-512	+256	-128	+0	+32	+0	+0	-4	+2	-1

Sum the decimal values up
-512+256-128+32-4+2-1 = -355

We can then convert out original number to check our work

Two's complement(10 1001 1101) = - 01 0110 0011

Place	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Binary	0	1	0	1	1	0	0	0	1	1
Decimal	+0	+256	+0	+64	+32	+0	+0	+0	+2	+1

-(256+64+32+2+1) = -355

This justifies the previous Booth's recoded number

B)

1010 * 0101 = 1110 0010   -6 * 5 = -30

MD	AC	MQ	MQ-1	Step	Iterations
1010	0000	0101	0		0
1010	0110	0101	0	AC-MD	1
1010	0011	0010	1	>>	1
1010	1101	0010	1	AC+MD	2
1010	1110	1001	0	>>	2
1010	0100	1001	0	AC-MD	3
1010	0010	0100	1	>>	3
1010	1100	0100	1	AC+MD	4
1010	1110	0010	0	>>	4

Final result is 1110 0010 which is –30 in decimal, -6 * 5 = 30

4. Perform the restoring division for [1011 / 0100], i.e., 11 / 4 in decimal, and show the contents of registers for each step. Also show the resulting quotient and remainder in binary numbers.

MD	AC	MQ
….	….	….

1011 / 0100 = Q:0010 R:0011   11 / 4 = Q:2 R:3

MD	AC	MQ	Step	Iterations
0100	0000	1011		0
0100	0001	0110	<<	1
0100	1101	0110	AC-MD	1
0100	1101	0110	MQ0=0	1
0100	0001	0110	RESTORE	1
0100	0010	1100	<<	2
0100	1110	1100	AC-MD	2
0100	1110	1100	MQ0=0	2
0100	0010	1100	RESTORE	2
0100	0101	1000	<<	3
0100	0001	1000	AC-MD	3
0100	0001	1001	MQ0=1	3
0100	0011	0010	<<	4
0100	1111	0010	AC-MD	4
0100	1111	0010	MQ0=0	4
0100	0011	0010	RESTORE	4

MQ = Q = 0010 = 2
AC = R = 0011 = 3
Check with definition of quotient remainder theorem
    A = (B * Q) + R
    11 = (4*2) + 3

5. Perform the non-restoring division for [1011 / 0100], i.e., 11 / 4 in decimal, and show the contents of registers for each step. Also show the resulting quotient and remainder in binary numbers.

MD	AC	MQ
….	….	….

1011 / 0100 = Q:0010 R:0011   11 / 4 = Q:2 R ☺

MD	AC	MQ	Step	Iterations
0100	0000	1011		0
0100	0001	0110	<<	1
0100	1101	0110	AC-MD	1
0100	1101	0110	MQ0=0	1
0100	1010	1100	<<	2
0100	1110	1100	AC+MD	2
0100	1110	1100	MQ0=0	2
0100	1101	1000	<<	3
0100	0001	1000	AC+MD	3
0100	0001	1001	MQ0=1	3
0100	0011	0010	<<	4
0100	1111	0010	AC-MD	4
0100	1111	0010	MQ0=0	4
0100	0011	0010	AC+MD	FINAL

MQ = Q = 0010 = 2
AC = R = 0011 = 3
Check with definition of quotient remainder theorem
    A = (B * Q) + R
    11 = (4*2) + 3

6. Explain briefly how the non-restoring division algorithm achieves higher efficiency than the restoring division algorithm.

The restoring division algorithm performs 3*B + R operations, where B is the number of bits of the inputs, and R is the number of restoring operations that have to be done (equal to B at most).
The non-restoring division algorithm performs 3*B + F operations, where B is the number of bits of the inputs, and F is the number of times the final operation runs (once at most).

Looking at these equations we can see that the number of operations for both equations in the best case (where there is no restoring operations at all) is equivalent. However, in the worst case for the restoring division algorithm there can be B times that the restore operation is performed. Compare this to the non-restoring division algorithm which performs an extra operation at the end in the worst case. This makes the non-restoring division algorithm much faster as it has an equivalent best case and a much faster worst case scenario.