CO224 LAB 5 PART 5

GROUP NO- 3

mult OPEARATION

ASSIGNED OPCODE- 00001101

ENCODING AND DECODING SAME AS A ADD OPERATION.

ALGORITHM USED FOR THE CALCULATION IS SHOWN BELOW. (THE SHOWN RESULT BELOW IS FOR TWO 4 BIT NUMBERS. IN THE CODE 2 NUMBERS OF 8-BIT EACH WERE MULTIPLED BUT THE LOGIC WAS THE SAME.

Here is the formal algorithm.

1. Initialize C = 0, Y = 0, X = multiplicand, Z = multiplier, Count = N.
2. At each step, examine Z_0.
   If $Z_0 = 1$, then add X and Y to put the sum in Y and set the carry bit C.
3. Right shift the register pair (Y, Z), with C $\rightarrow Y_{N-1}$ and $Y_0 \rightarrow Z_{N-1}$. Z_0 is lost.
4. Decrement the count. If count = 0, stop. If not, go to step 2.

We now illustrate the algorithm with the example specified just above.

At the start of the multiplication, the registers are initialized as follows:
X = 1011, Y = 0000, Z = 1101, and C = 0.

$Z_0 = 1$, so there is an addition.

Then there is a right shift.

Now $Z_0 = 0$, so there is no addition. The next step is another right shift.

$Z_0 = 1$, so there is an addition.

$Z_0 = 1$, so there is an addition.

```
  1011
  0010
 01101
```

Here, the Y register gets the 4–bit sum 1101, and the carry bit is set to C = 0.

Then, there is the third right shift.

$Z_0 = 1$, so there is an addition.

```
  1011
  0110
 10001
```

Here, the Y register gets the 4–bit sum 0001, and the carry bit is set to C = 1.

The final shift gives the product 1000 1111 (decimal 143) in the (Y, Z) register pair.

NO CHANGES WERE MADE TO THE DATAPATH FOR THIS INSTRUCTION.

EX OF A ASSEMBLY CODE TO MACHINE INSTRUCTION

mult 6 4 3 -> 00001101 00000110 00000100 00000011

sll/srl OPEARATIONS

ASSIGNED OPCODEs-

sll- 00001000

srl- 00001001

HERE in both sll and srl, the operations are perfomed on a 8 bit number. so if an 8 bit number is logically shifted by more than 8 places it becomes zero. because of that the immediate value which is of 8 bits has redundant bits as only 4 bits are needed to fully define all the logic shift operations. because of that the 8^{th} bit of the immediate value is taken to store information about the type of logical shift. if the 8^{th} bit is 1 it is a right shift operation and if it is 0 then it is a left shift operation. THIS ALLOWED BOTH OPERATIONS TO BE CONDUCTED USING JUST ONE ALUOP

so if a left shift operation is to be performed to shift left by 2 places the immediate shift value is encoded as - 00000010

if a right shift operation is to be performed to right left by 2 places the immediate shift value is encoded as - 10000010

EX OF A ASSEMBLY CODE TO MACHINE INSTRUCTION

sll 2 1 0x02 -> 00001000 00000010 00000001 00000010

srl 6 2 0x02-> 00001001 00000110 00000010 10000010

sra/ror OPEARATIONS

ASSIGNED OPCODEs-

sra- 00001010

ror- 00001011

NO CHANGES WERE MADE TO THE DATAPATH. BOTH IMEPLEMENTED USING THE SAME FUNCTIONAL GATE IMPLEMENTATION.

EX OF A ASSEMBLY CODE TO MACHINE INSTRUCTION

ror 7 4 0x02 -> 00001011 00000111 00000100 00000010

sra 7 5 0x02 -> 00001010 00000111 00000101 00000010

bne OPEARATIONS

ASSIGNED OPCODE

bne- 00001100

A NEW CONTROL SIGNAL BNQ WAS ADDED TO IMPLEMENT BRANCH IF NOT EQUAL INSTRUCTION. VERY SIMILAR TO BRANCH IF EQUAL INSTRUCTION

EX OF A ASSEMBLY CODE TO MACHINE INSTRUCTION

bne 0x02 4 6 -> 00001100 00000010 00000100 00000110