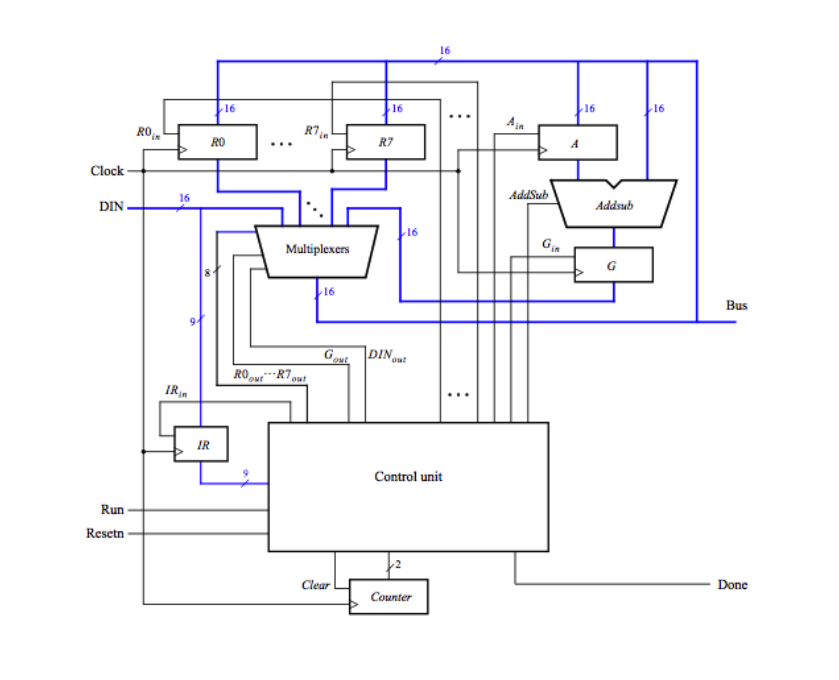
PROCESSOR 2014

John Butler and Antonia Lewis |CSE 2304

## 1. OVERVIEW

 Our design was based off of the diagram shown above depicting a 16-bit processor with 8 registers. From this image we have constructed our processor using the VHSIC Hardware Description Language (VHDL), consisting of a total of 11 separate component files. Major components include an arithmetic logic unit (ALU), counter, control unit and a 10 to 1 multiplexer that all contribute to one main file (PROCESSOR.VHD). With these entities the processor allows for adding, subtracting, and moving data between registers by using a 4 step clock to distinguish between instruction cycles.

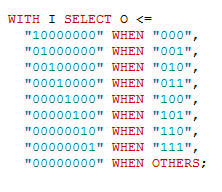
## 2. INSTRUCTIONS & DATAFLOW

The processor accepts a 16-bit input (DIN), which can contain either 9-bit instructions (discarding the final seven bits) or a 16-bit immediate. The processor reuses its 16-bit data input for carrying instructions: the first nine bits are sent to the instruction register when appropriate. These instructions are broken up into three pieces: three bits for the instruction identifier, three bits for the location of the first register, and three bits for the location of the second register.

## IIIXXXYYY

INTRUCTION REGISTER X REGISTER Y

Evaluating a given instruction can take up to four clock cycles, starting with the initial 00 time step in which the instruction is read from the input stream and placed into the IR register. From there, the instruction is fed to the control unit, where the register numbers XXX and YYY will be converted into individual enables.



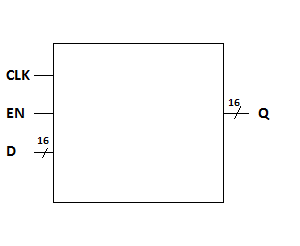
The decoder follows the logic shown above in order to turn 3 bits from an instruction into an 8 bit vector for deciding the enables. This process is done twice for bits 3-5 (RX) and 6-8 (RY) of the instruction. These decoded outputs are used for calculating when to write to the indicated registers based off of which bit has been turned on (1). Since each instruction consists of a different operation during each clock cycle, these enables are updated on each cycle in order to assure the correct registers are currently enabled for reading or writing. Once all of the enables have been calculated the processor proceeds as indicated by the current instruction.

The instructions themselves are enumerated in the following table. For each time step and instruction, the control unit will assert a different combination of inputs. These will control the behavior of each other component in the processor.

|  |  |
| --- | --- |
| Instruction & Parameters | Operation Performed |
| MV R­x, R­­y | R­x 🡨 [R­­y] |
| MVI R­x, #D | R­x 🡨 D |
| ADD R­x, Ry | R­x 🡨 [R­x] + [R­­y] |
| SUB R­x, Ry | R­x 🡨 [R­x] - [R­­y] |

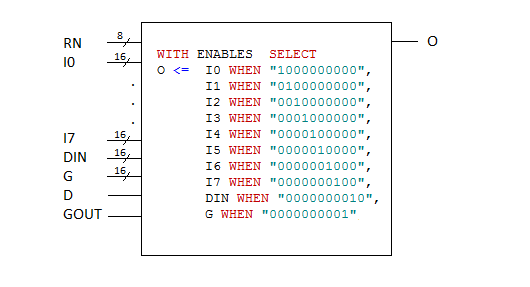
## 3. MAJOR COMPONENTS

### REGISTER\_16.VHD



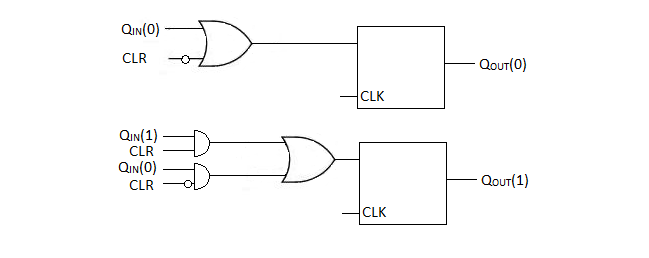
All of the registers within the processor are constructed using a series of D flip flops, as shown in the above example of the 16 bit register used for storing data. When enabled and a positive clock edge is encountered, the 16 bit input D is stored into Q. The same idea is used for the 9 bit instruction register.

### MULTIPLEXER 10 TO 1



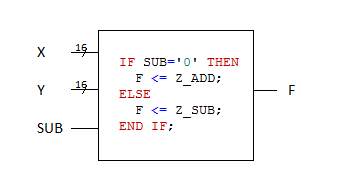
The multiplexer is used to select between ten different 16-bit inputs: the output from each of the main registers R0…R7, plus the output from the A and G registers (used for arithmetic operations). The appropriate data stream is chosen by a 10-bit vector, formed by concatenating the register enable vector (RN) with the DIN\_OUT and G\_OUT enables.

### COUNTER

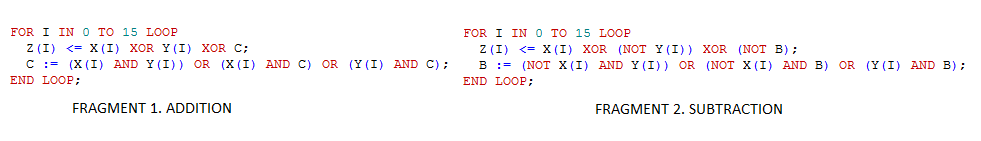


The counter is a vital component that indicates which step of an instruction the processor is currently carrying out. The component itself takes in a clear signal, clock signal and a 2 bit vector indicating its previous time-step. After being processed by two T flip-flops the counter is either incremented by one step or cleared based off of the inputs. Instructions take varying numbers of clock cycles, so if less than four cycles are needed, the processor will manually reset the counter using the CLR.

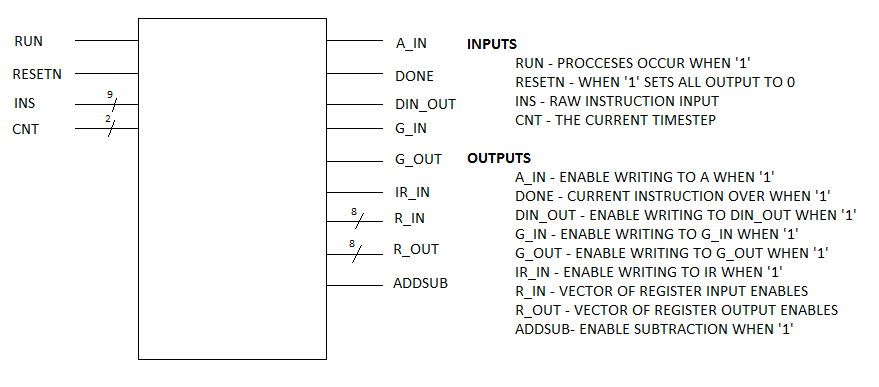
### Arithmetic Logic Unit (ALU)



The processor contains logic for two unsigned mathematical operations, addition and subtraction. The ALU takes in two 16 bit operands and an input ‘SUB’ which if enabled will send the information to the subtractor entity else the numbers are sent to the adder. Addition and subtraction are both handled using the ripple adder paradigm; the value of each bit position is computed individually, and then the appropriate carry/borrow bit is sent to the next digit.



### Control unit



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | T0 (00) | T1 (01) | T2 (10) | T3 (11) |
| MV (000) | *IR\_IN* | *RY\_OUT*  *RX\_IN*  *DONE* |  |  |
| MVI (001) | *IR\_IN* | *RX\_IN*  *DIN\_OUT* | *DONE* |  |
| ADD (010) | *IR\_IN* | *RX\_OUT*  *A\_IN* | *RY\_OUT*  *G\_IN* | *RX\_IN*  *G\_OUT*  *DONE* |
| SUB (011) | *IR\_IN* | *RX\_OUT*  *A\_IN* | *RY\_OUT*  *G\_IN*  *ADDSUB* | *RX\_IN*  *G\_OUT*  *DONE* |

Above is a diagram of the various inputs and outputs of the control unit. This entity is the heart of our processor, taking in and directing traffic based off of the decoded instruction and a series of conditional (concurrent) signal assignment statements. These statements look at the current time step of the counter and the instruction in order to set the enables as needed. Below is a table indicating which outputs are enabled during which instruction and time step.

## 4. DEMONSTRATION

### move immediate (I = 001)

### 

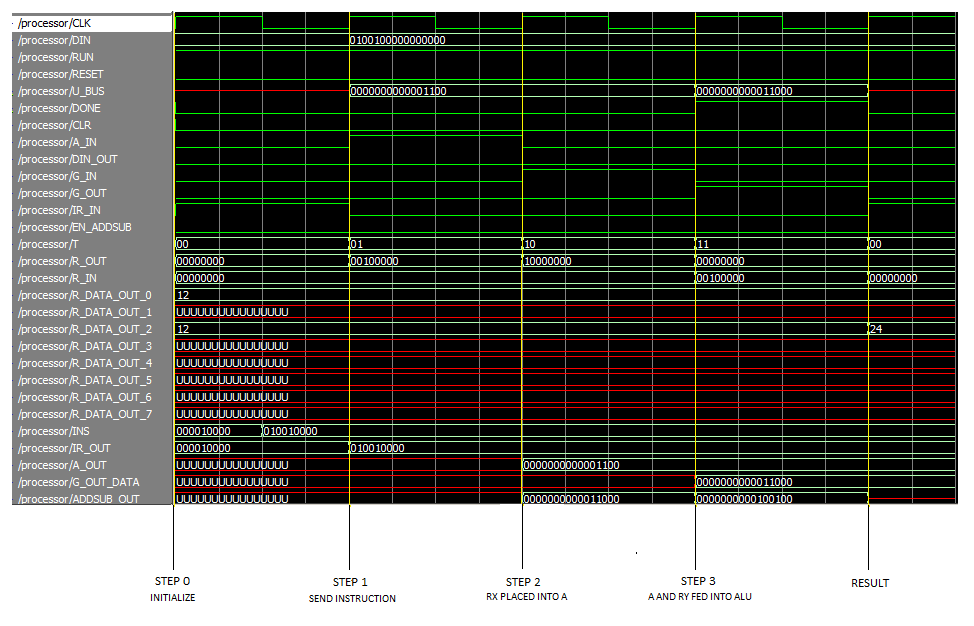
Once the move immediate instruction is placed into the IR register (Shown above in step 2) the counter increments to the next clock cycle and can accept the 16-bit immediate data from DIN. Above we have selected to place the immediate 12 (0000000000001100) into the fourth register as indicated by the instruction. When DIN is input it is sent to the bus and placed into the appropriate register based off of the enables generated by the control unit. The process finishes at time step 2 and would reset on the next rising clock edge; that is, the CLR signal is asserted at this time and the counter would subsequently reset to 0 in the following step.

### move from register (I = 000)

### 8

In order to copy from one register to another there is the instruction MOV. Starting with the clock at 00 the instruction is input through DIN and sent into the IR register. During this step the data from the indicated RY register is placed on the bus. On the next rising edge of the clock the data is placed from the bus into the into the specified RX register. This operation only requires one clock cycle (in addition to the ubiquitous IR\_IN step), so the CLR signal is asserted at the same time.

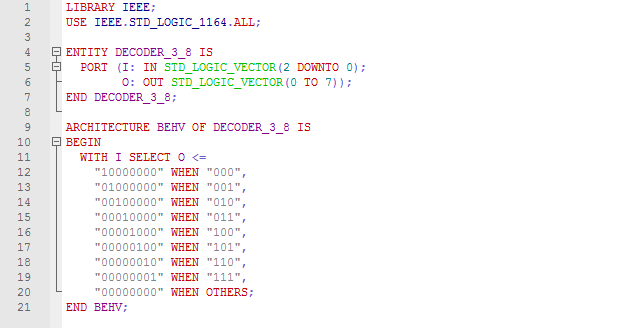
### addition & subtraction (010 & 011)



The processor handles the execution of moving the operands of both addition and subtraction in the same way. The instruction carries information on whether or not to toggle the subtraction enable, a destination/first operand register RX and a second operand RY. In the first clock cycle RX is placed onto the bus and moved into the A register. After another rising edge RY is placed onto the bus and fed into the ALU along with the contents of the A register. In the final step the ALU places the result of the operation into the G\_OUT signal, and this is placed onto the bus via the multiplexer; from there it is written into the appropriate register. Note that these operations actually require the full four clock cycles to complete, due to the additional data transfer involving the A and G registers, and so the CLR signal is not required to reset the counter to zero.

## 5. APPENDIX

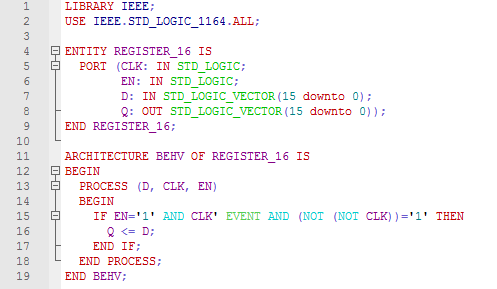
### decoder\_3\_8.vhd



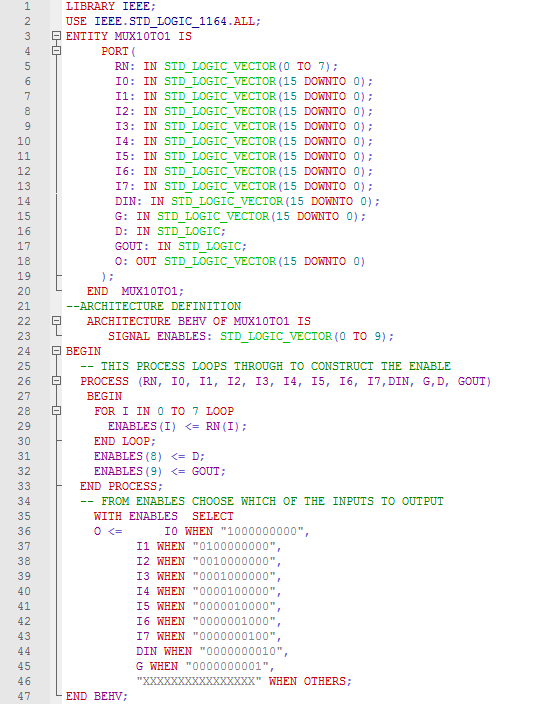
### REGISTER\_9.VHD

### 

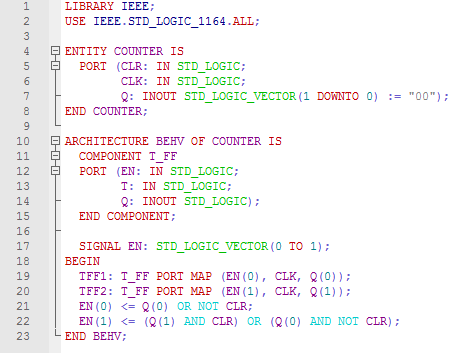
### REGISTER\_16.VHD



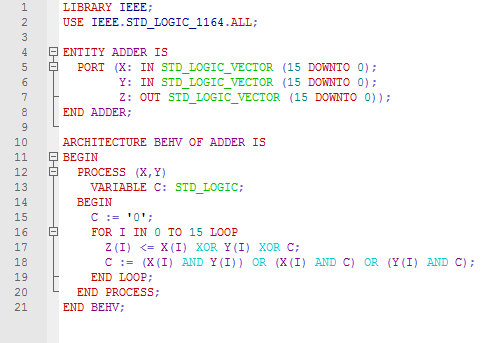
MUX\_10\_1.VHD



COUNTER.VHD



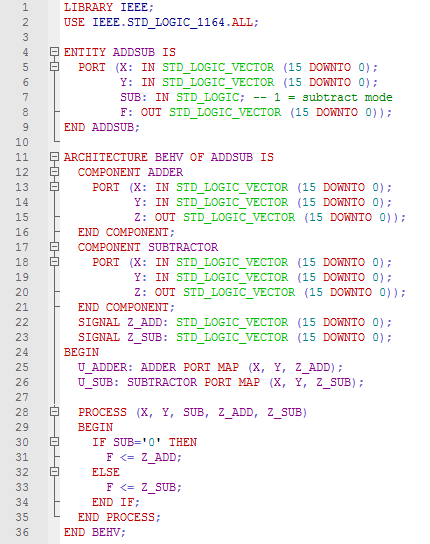
### ADDER.VHD



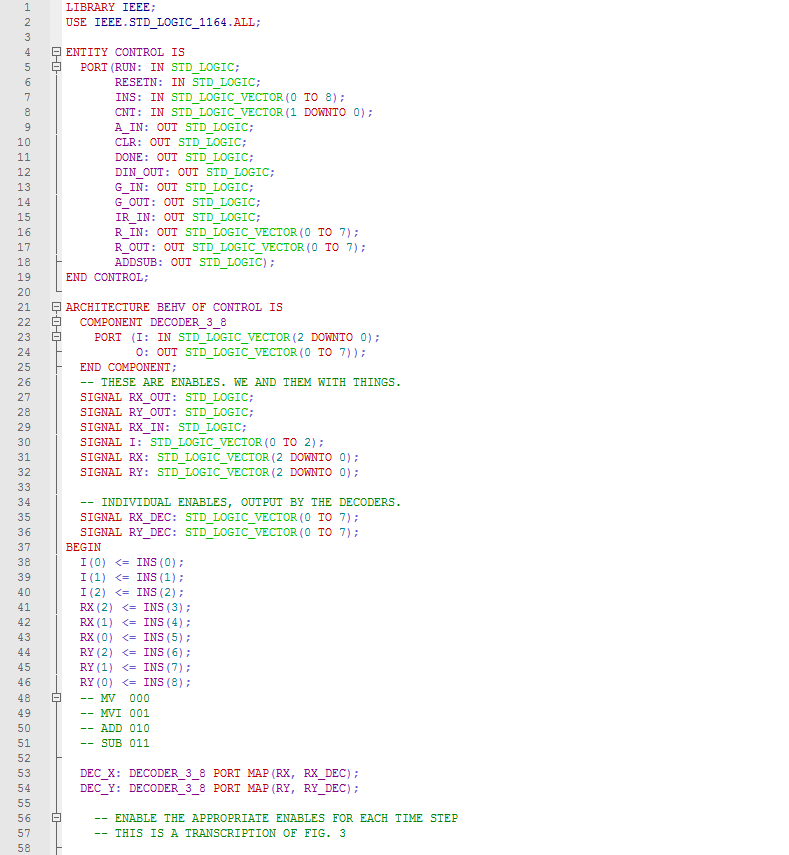
### SUBTRACTOR.VHD

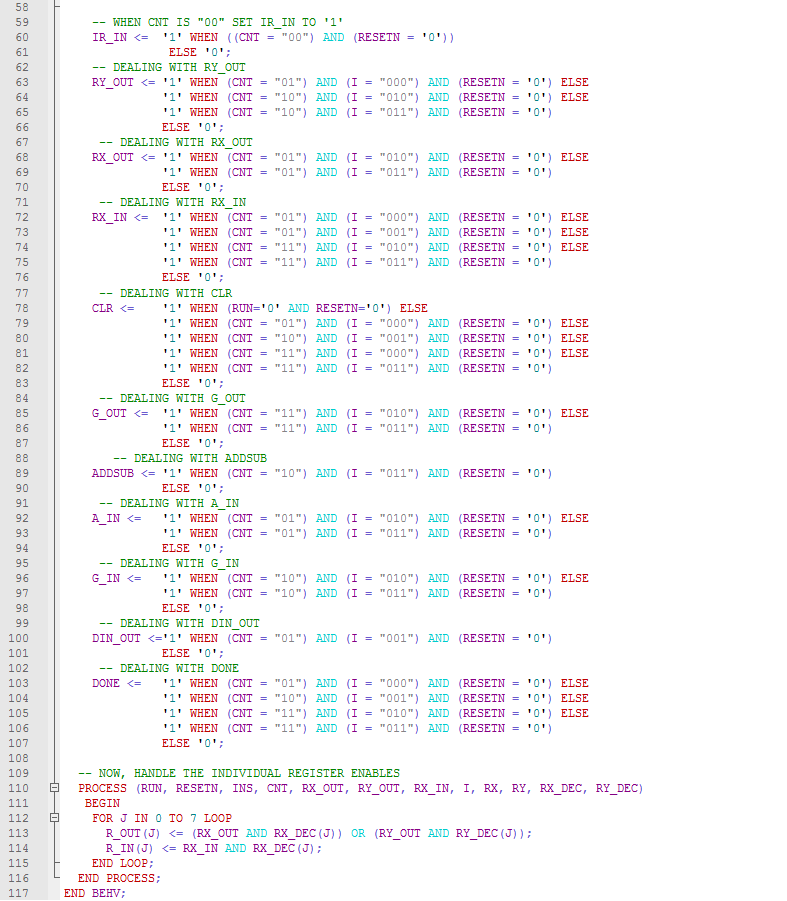
### 

### ADDSUB.VHD



### CONTROLUNIT.VHD





### PROCESSOR.VHD

