Final Project

Darren Codipilly, Brandon Quant

COEN 122

Wednesday 2:15-5:00

Abstract:
For this project we created a datapath that takes in custom assembly instructions and converts those instructions into machine language. The components of the project include muxes, ALUs, the program counter, the instruction memory, the register file, the sign extender, the control unit, the data memory, and the various gates and buffers.

Description of the CPU:
For the instruction fetch stage, we start with the mux and based on the selects of the mux, the 32 bit value is then passed to the program counter. The program counter contains the address or location of the instruction being executed at the current time. Then the 8 bit output is then the input for the first ALU, the instruction memory, and the IF/ID buffer. The instruction memory holds the instructions that need to be decoded and executed. Then the 32 output of the instruction memory is then also passed as the input for the ID/IF buffer. The IF/ID connects the instruction fetch and the instruction decode stages, where the instruction memory and program counter are passed through.

For the instruction decode stage, the instruction memory is 32 bits, where 2 sets of six bits represent the registers that are inputs to the register file. The write address input comes from the write back stage for the rd register. The dataIn and the write inputs come from the memory access stage. The 4 most significant bits of the instruction memory would be the opcode and input to the control unit, where the control unit sets the flags and the aluOp. The 22 least significant bits of the instruction memory that are passed through the IF/ID buffer are extended to 32 bits in the sign extender. This output, along with the program counter output that is passed through the IF/ID buffer passes through the second ALU, where this output is then passed to the ID/EX buffer. All the outputs of this stage are passed through as inputs of the ID/EX buffer.

For the execution stage, the outputs from the register file that are passed through the ID/EX buffer are the inputs for the data memory. The rd register input is passed through as an input for the EX/WB buffer. The flags of MemRead and MemWrite are inputs into the data memory and the ALUop is the select for the third ALU. The rest of the flags are passed through as inputs to the EX/WB buffer. The inputs of the third ALU would be the outputs of the register file that is passed through the ID/EX buffer. The outputs of the third ALU as well as the N and Z flags are passed through the EX/WB buffer. Also, the output of the second ALU passes straight through to the EX/WB buffer.

For the memory/write back stage, we have 2 AND gates, where the first AND gates' inputs are the N and Branch Neg flags while the second AND gates' inputs are the Z and Branch Zero flags. The output of these 2 AND gates are inputs to an OR gate which also takes in the input of the jump flag. This output is then one of the selects for the first MUX in the instruction fetch

stage. The inputs of the second mux would be the third ALU"s output, the data memory output and the second ALU's output which are all passed through from the EX/WB buffer. The selects for the mux are MemtoReg and PCtoReg flags. The output of this second mux is the input is data input for the register file. The outputs of the third ALU and the data memory become the inputs for the first mux. The rd output becomes the write address for the register file in the instruction decode stage.

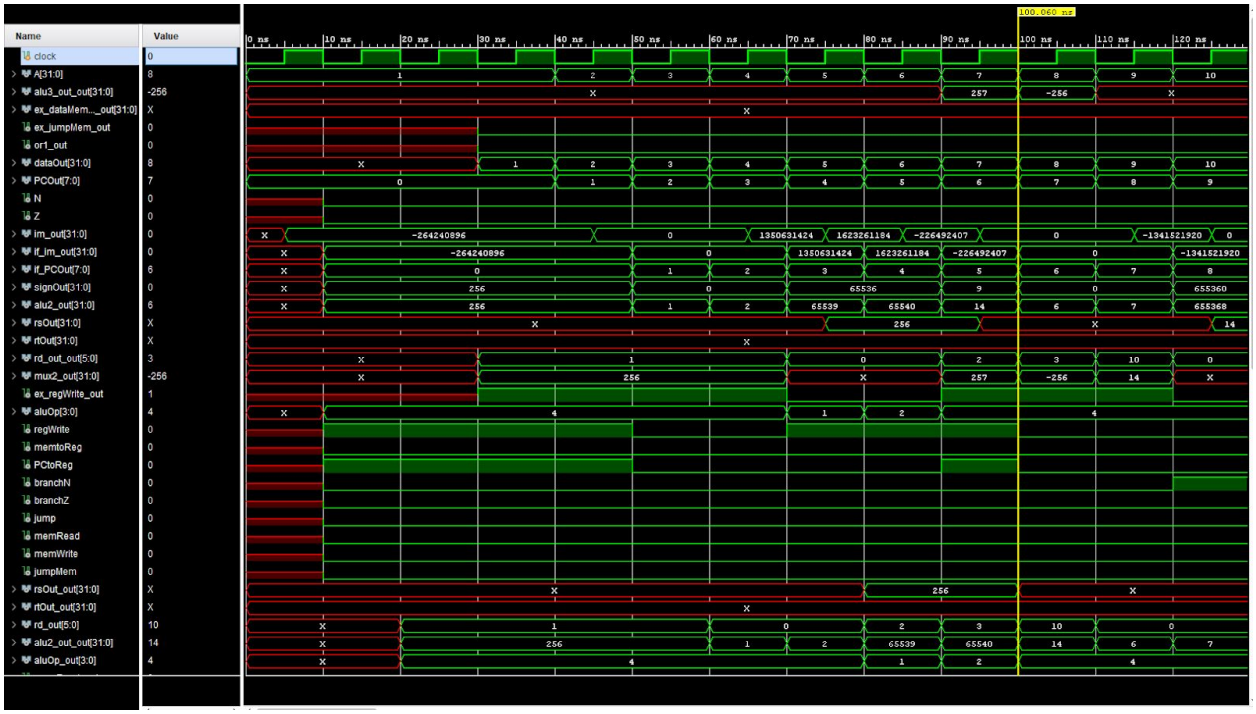
Truth Table for Control:

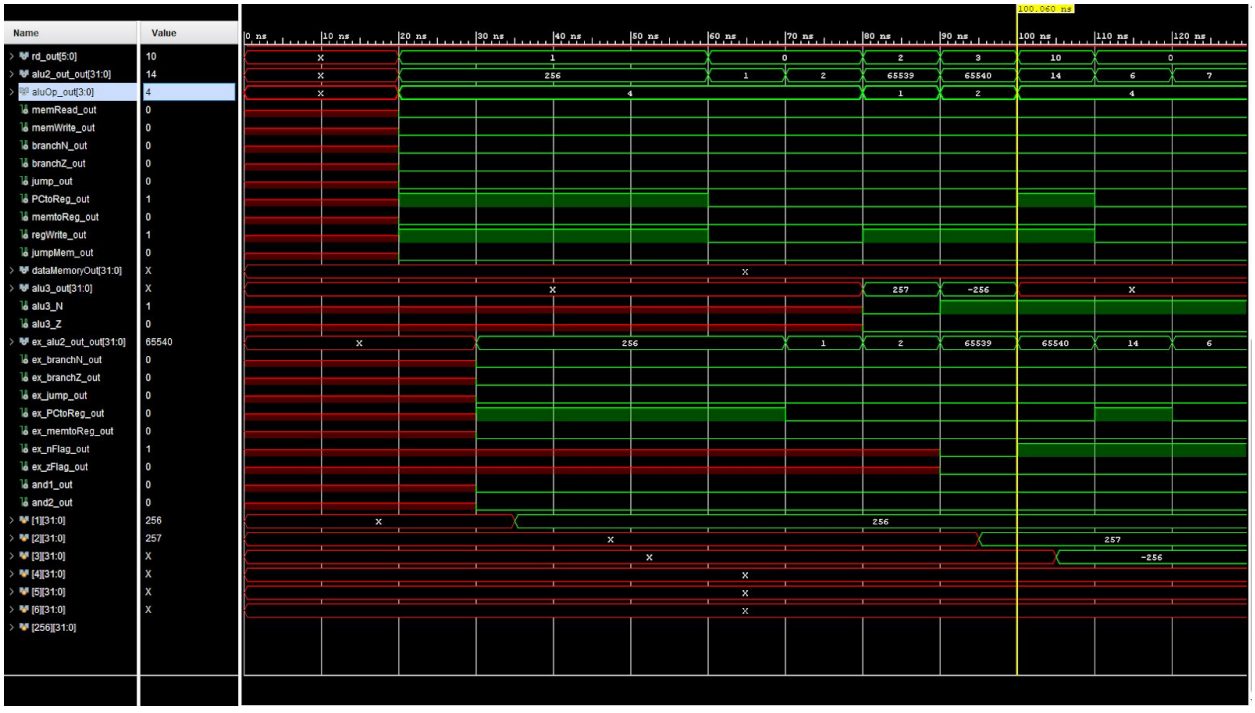
Op-code	**regWrite**	**memtoReg**	**PCtoReg**	**branchN**	**branchZ**	**jump**	**jumpMem**	**memRead**	**memWrite**	**aluOp**
0000	0	0	0	0	0	0	0	0	0	0100
1111	1	0	1	0	0	0	0	0	0	0100
1110	1	1	0	0	0	0	0	1	0	0100
0011	0	0	0	0	0	0	0	0	1	0100
0100	1	0	0	0	0	0	0	0	0	0000
0101	1	0	0	0	0	0	0	0	0	0001
0110	1	0	0	0	0	0	0	0	0	0010
0111	1	0	0	0	0	0	0	0	0	0011
1000	0	0	0	0	0	1	0	0	0	0100
1001	0	0	0	0	1	0	0	0	0	0100
1010	0	0	0	0	0	0	1	1	0	0100
1011	0	0	0	1	0	0	0	0	0	0100

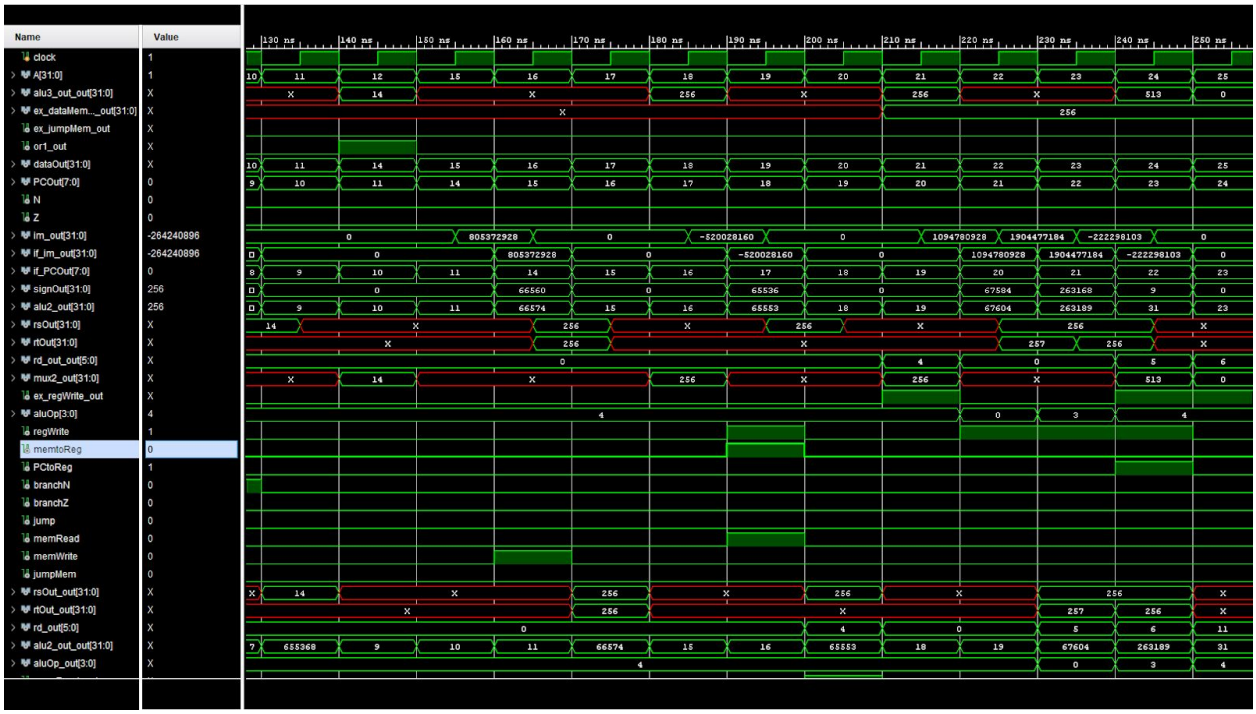
Summation Code:

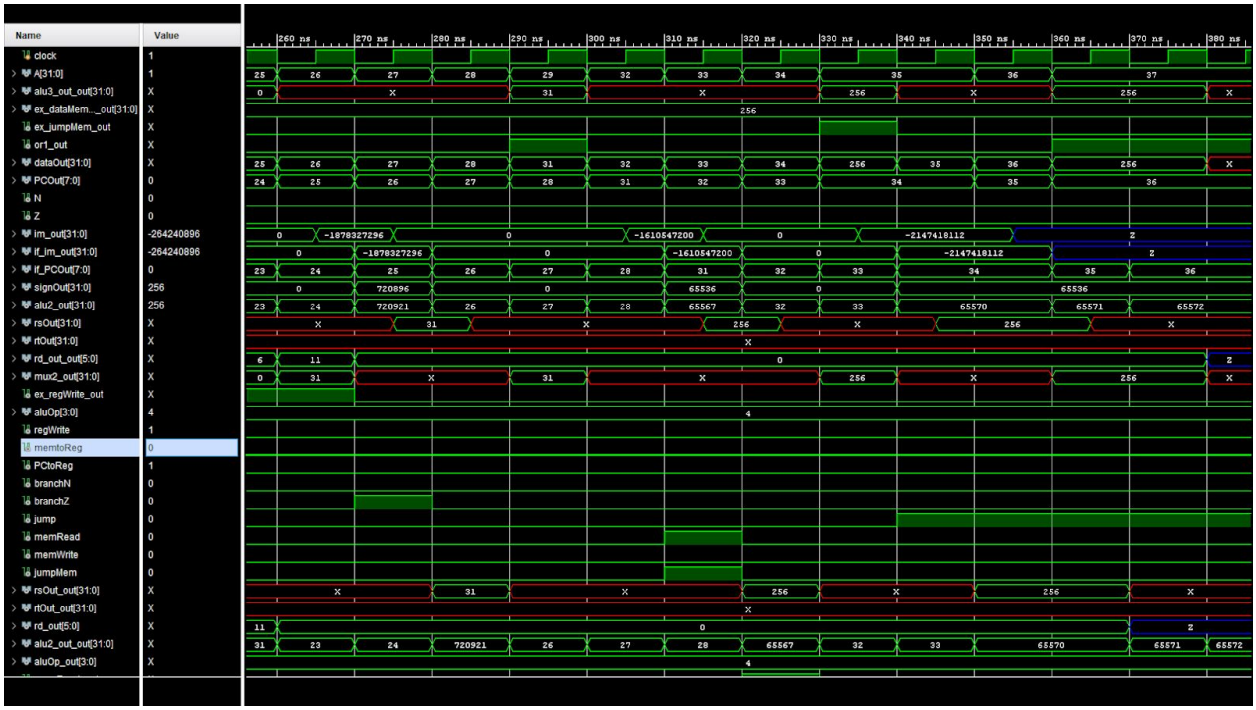
```
//Assume x0 = 0, x1 = rs, x2 = rt, x7 = 3, x8 = 1,  x9 = rd

ADD x3, x1, x2
SUB x3, x3, x8 //x3 = rs+rt-1
LDPC x5, 7

 SUB x4, x3,x1 //(rs-rt-1)-i
 BRZ x5
 LD x6, [x1] //x6 = memory[i]
 ADD x0, x0, x6
 INC x1, x1
 J x7

 STUR x0, x9
```

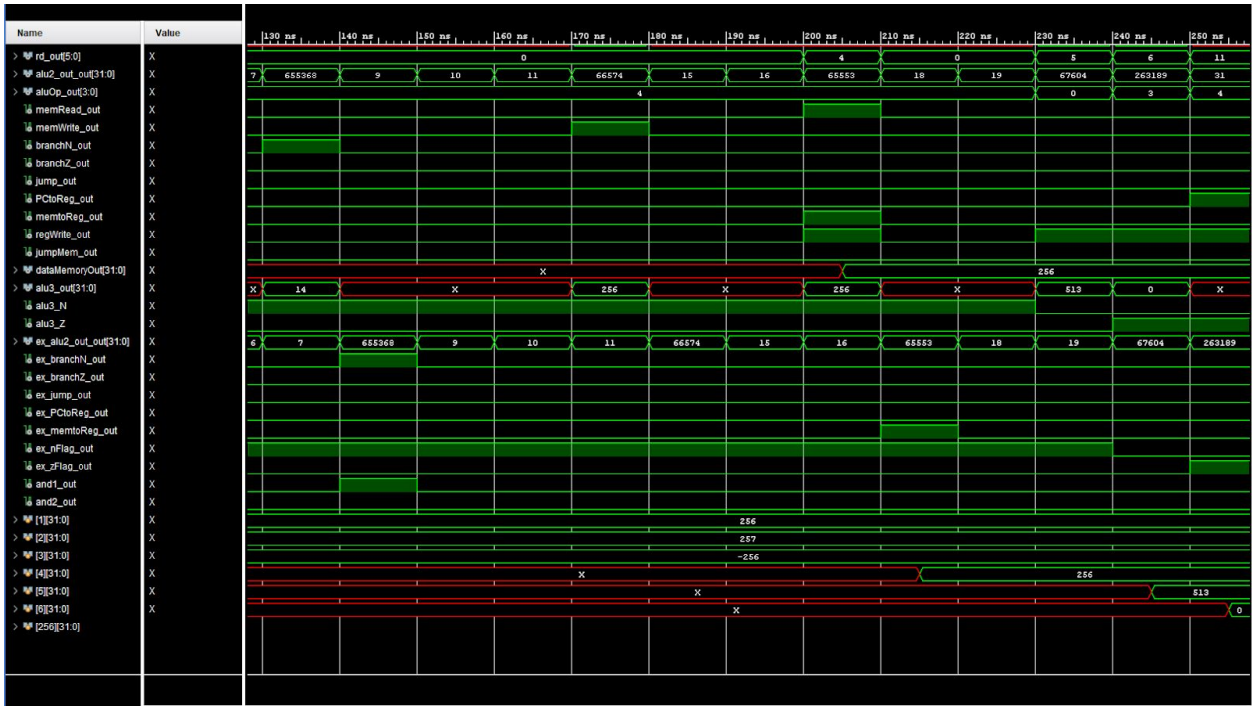
Waveforms:

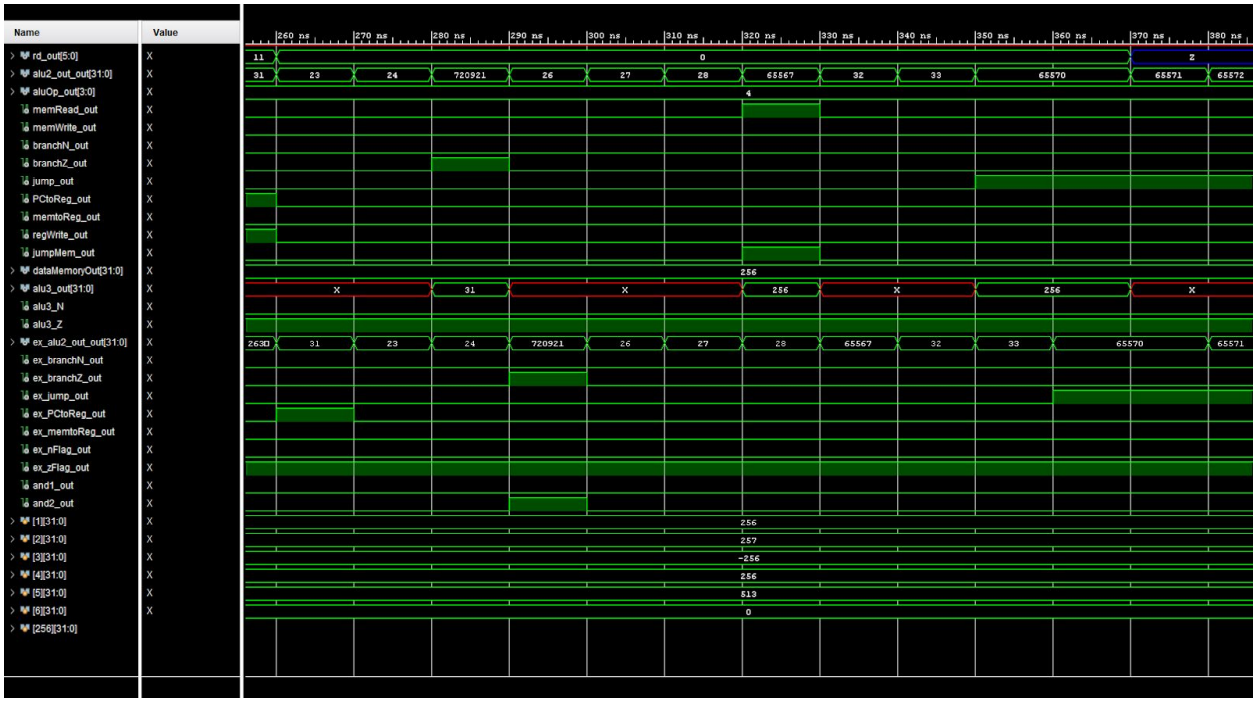

Name
Value
clock
A[31:0]
alu3_out_out[31:0]
ex_dataMem..._out[31:0]
ex_jumpMem_out
or1_out
dataOut[31:0]
PCOut[7:0]
N
Z
im_out[31:0]
If_Im_out[31:0]
If_PCOut[7:0]
signOut[31:0]
alu2_out[31:0]
rsOut[31:0]
rtOut[31:0]
rd_out_out[5:0]
mux2_out[31:0]
ex_regWrite_out
aluOp[3:0]
regWrite
memtoReg
PCtoReg
branchN
branchZ
jump
memRead
memWrite
jumpMem
rsOut_out[31:0]
rtOut_out[31:0]
rd_out[5:0]
alu2_out_out[31:0]
aluOp_out[3:0]

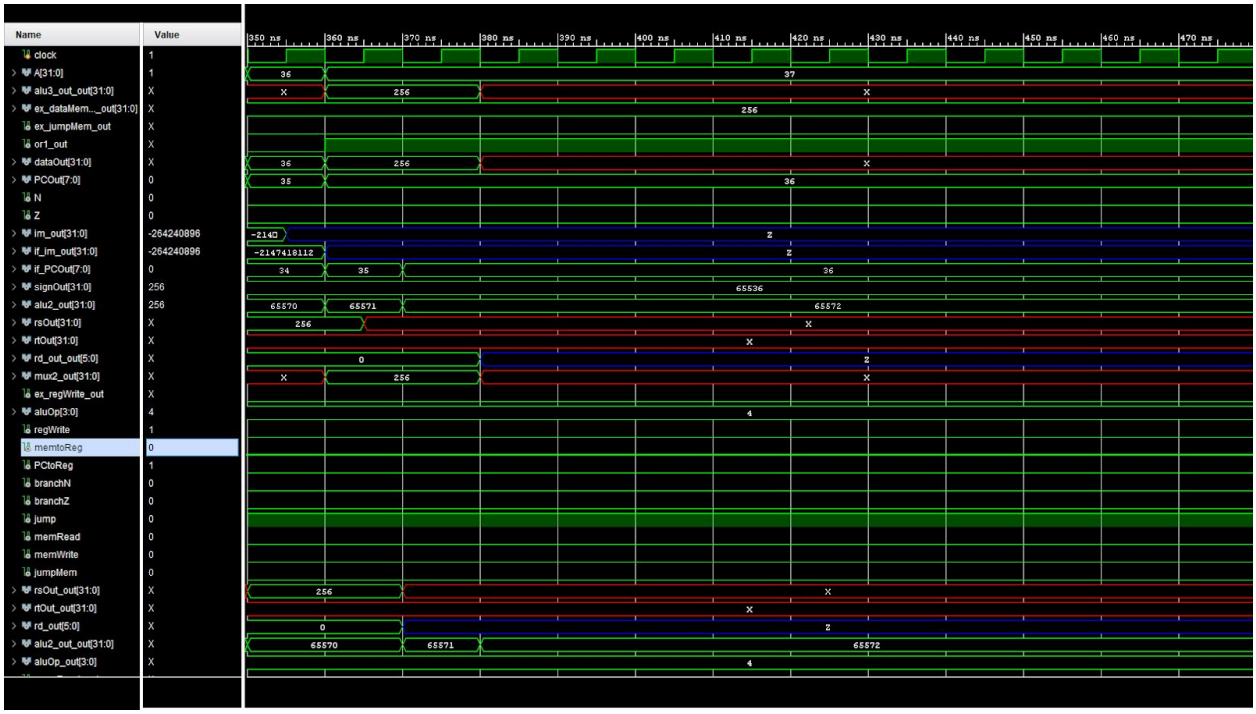

Name
Value
rd_out[5:0]
alu2_out_out[31:0]
aluOp_out[3:0]
memRead_out
memWrite_out
branchN_out
branchZ_out
jump_out
PCtoReg_out
memtoReg_out
regWrite_out
jumpMem_out
dataMemoryOut[31:0]
alu3_out[31:0]
alu3_N
alu3_Z
ex_alu2_out_out[31:0]
ex_branchN_out
ex_branchZ_out
ex_jump_out
ex_PCtoReg_out
ex_memtoReg_out
ex_nFlag_out
ex_zFlag_out
and1_out
and2_out
[1][31:0]
[2][31:0]
[3][31:0]
[4][31:0]
[5][31:0]
[6][31:0]
[256][31:0]


Name
Value
clock
A[31:0]
alu3_out_out[31:0]
ex_dataMem..._out[31:0]
ex_jumpMem_out
or1_out
dataOut[31:0]
PCOut[7:0]
N
Z
im_out[31:0]
if_im_out[31:0]
if_PCOut[7:0]
signOut[31:0]
alu2_out[31:0]
rsOut[31:0]
rtOut[31:0]
rd_out_out[5:0]
mux2_out[31:0]
ex_regWrite_out
aluOp[3:0]
regWrite
memtoReg
PCtoReg
branchN
branchZ
jump
memRead
memWrite
jumpMem
rsOut_out[31:0]
rtOut_out[31:0]
rd_out[5:0]
alu2_out_out[31:0]
aluOp_out[3:0]
130 ns
140 ns
150 ns
160 ns
170 ns
180 ns
190 ns
200 ns
210 ns
220 ns
230 ns
240 ns
250 ns


Name
Value
clock
A[31:0]
alu3_out_out[31:0]
ex_dataMem..._out[31:0]
ex_jumpMem_out
or1_out
dataOut[31:0]
PCOut[7:0]
N
Z
im_out[31:0]
if_im_out[31:0]
if_PCOut[7:0]
signOut[31:0]
alu2_out[31:0]
rsOut[31:0]
rtOut[31:0]
rd_out_out[5:0]
mux2_out[31:0]
ex_regWrite_out
aluOp[3:0]
regWrite
memtoReg
PCtoReg
branchN
branchZ
jump
memRead
memWrite
jumpMem
rsOut_out[31:0]
rtOut_out[31:0]
rd_out[5:0]
alu2_out_out[31:0]
aluOp_out[3:0]
260 ns
270 ns
280 ns
290 ns
300 ns
310 ns
320 ns
330 ns
340 ns
350 ns
360 ns
370 ns
380 ns


Name
Value
rd_out[5:0]
alu2_out_out[31:0]
aluOp_out[3:0]
memRead_out
memWrite_out
branchN_out
branchZ_out
jump_out
PCtoReg_out
memtoReg_out
regWrite_out
jumpMem_out
dataMemoryOut[31:0]
alu3_out[31:0]
alu3_N
alu3_Z
ex_alu2_out_out[31:0]
ex_branchN_out
ex_branchZ_out
ex_jump_out
ex_PCtoReg_out
ex_memtoReg_out
ex_nFlag_out
ex_zFlag_out
and1_out
and2_out
[1][31:0]
[2][31:0]
[3][31:0]
[4][31:0]
[5][31:0]
[6][31:0]
[256][31:0]
130 ns
140 ns
150 ns
160 ns
170 ns
180 ns
190 ns
200 ns
210 ns
220 ns
230 ns
240 ns
250 ns


Name
Value
rd_out[5:0]
alu2_out_out[31:0]
aluOp_out[3:0]
memRead_out
memWrite_out
branchN_out
branchZ_out
jump_out
PCtoReg_out
memtoReg_out
regWrite_out
jumpMem_out
dataMemoryOut[31:0]
alu3_out[31:0]
alu3_N
alu3_Z
ex_alu2_out_out[31:0]
ex_branchN_out
ex_branchZ_out
ex_jump_out
ex_PCtoReg_out
ex_memtoReg_out
ex_nFlag_out
ex_zFlag_out
and1_out
and2_out
[1][31:0]
[2][31:0]
[3][31:0]
[4][31:0]
[5][31:0]
[6][31:0]
[256][31:0]
260 ns
270 ns
280 ns
290 ns
300 ns
310 ns
320 ns
330 ns
340 ns
350 ns
360 ns
370 ns
380 ns


Name
Value
clock
A[31:0]
alu3_out_out[31:0]
ex_dataMem..._out[31:0]
ex_jumpMem_out
or1_out
dataOut[31:0]
PCOut[7:0]
N
Z
im_out[31:0]
if_im_out[31:0]
if_PCOut[7:0]
signOut[31:0]
alu2_out[31:0]
rsOut[31:0]
rtOut[31:0]
rd_out_out[5:0]
mux2_out[31:0]
ex_regWrite_out
aluOp[3:0]
regWrite
memtoReg
PCtoReg
branchN
branchZ
jump
memRead
memWrite
jumpMem
rsOut_out[31:0]
rtOut_out[31:0]
rd_out[5:0]
alu2_out_out[31:0]
aluOp_out[3:0]
350 ns
360 ns
370 ns
380 ns
390 ns
400 ns
410 ns
420 ns
430 ns
440 ns
450 ns
460 ns
470 ns


Name
Value
rd_out[5:0]
alu2_out_out[31:0]
aluOp_out[3:0]
memRead_out
memWrite_out
branchN_out
branchZ_out
jump_out
PCtoReg_out
memtoReg_out
regWrite_out
jumpMem_out
dataMemoryOut[31:0]
alu3_out[31:0]
alu3_N
alu3_Z
ex_alu2_out_out[31:0]
ex_branchN_out
ex_branchZ_out
ex_jump_out
ex_PCtoReg_out
ex_memtoReg_out
ex_nFlag_out
ex_zFlag_out
and1_out
and2_out
[1][31:0]
[2][31:0]
[3][31:0]
[4][31:0]
[5][31:0]
[6][31:0]
[256][31:0]
350 ns
360 ns
370 ns
380 ns
390 ns
400 ns
410 ns
420 ns
430 ns
440 ns
450 ns
460 ns
470 ns

Runtime Evaluation of CPU

Theoretical

Runtime of CPU = 1(delay of instruction memory) + 1(delay of data memory) + 1(delay of register file) + 3(delay of ALU)
= 1(2ns) + 1(2ns) + 1(1.5ns) + 3(2ns) = 10.5 ns

10.5ns * 36 instructions = **378 ns**

Actual Runtime:

Total runtime = **360 ns**