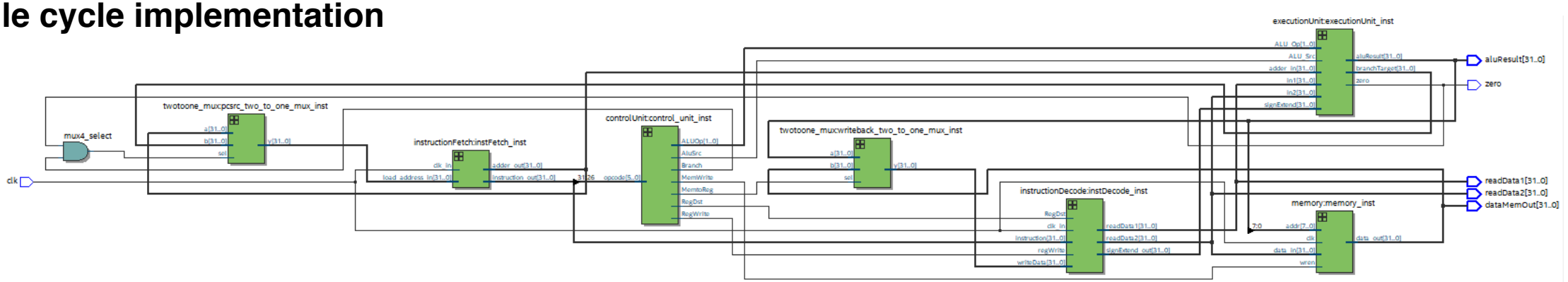
FUNDAMENTALS OF COMPUTER DESIGN: FINAL PROJECT

Nathan Gardner

Spring 2022 **ECE 5120**

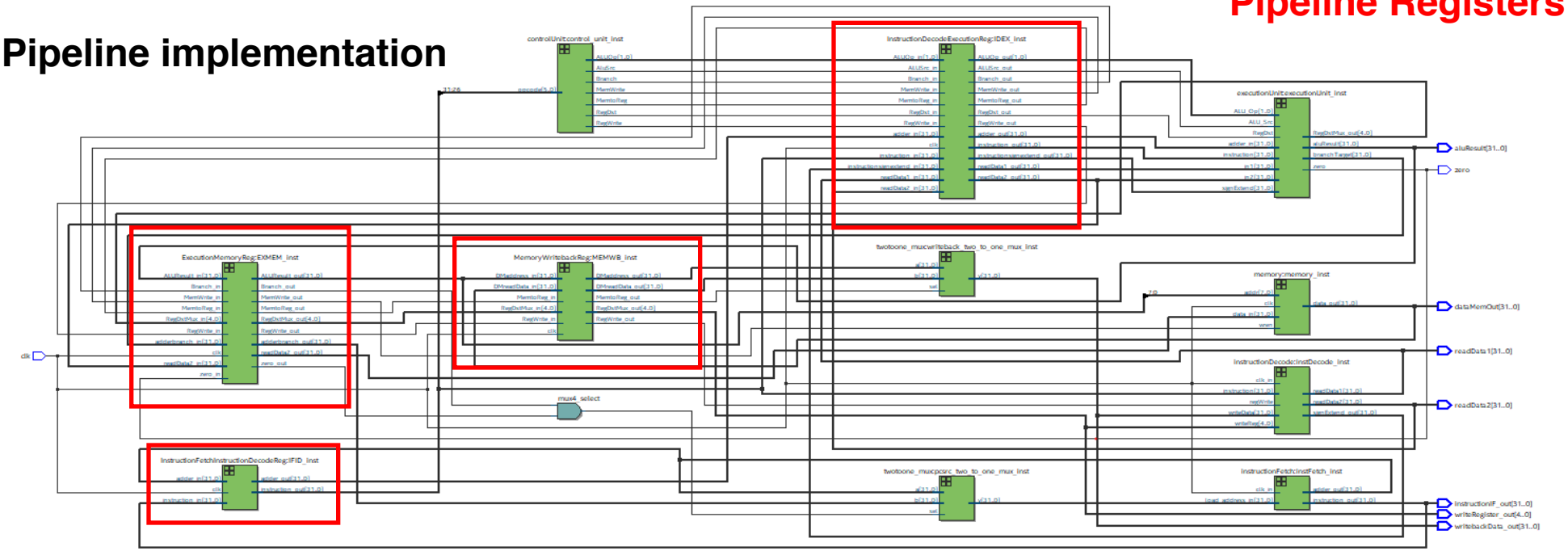
Successful implementation of single cycle and pipelined MIPS architecture design

Single cycle implementation



Pipeline implementation

Pipeline Registers



Nathan Gardner

Tennessee TECH

Register/ Memory Loading

- Registers were loaded in the the instruction memory with addi instructions
  - addi $t1, $zero, 0x112
  - addi $t2, $zero, 0xA
  - addi $t3, $zero, 0xF
- Memory loading was done using memory initialization files (.mif) files

```
CONTENT BEGIN
    00   :   00100000000010010000000100010010;--load phase addi $t1, $zero, 0x112, so that $t1 = 0x112
    01   :   00100000000010100000000000001010;--load phase addi $t2, $zero, 0xA, so that $t2 = 0xA
    02   :   00100000000010110000000000001111;--load phase addi $t3, $zero, 0xF, so that $t3 = 0xF
    03   :   00010001001010100000000000000011;--beq $t1, $t2, Equal
    04   :   00000001001010100100100000100000;--add $t1, $t1, $t2
    05   :   10101101010010110000000001100100;--sw $t3, 100($t2)
    06   :   00000001001010100100100000100101;--or $t1, $t1, $t2
    07   :   00000000000000000000000000000000;
    08   :   00000000000000000000000000000000;
    09   :   00000000000000000000000000000000;
    0A   :   00000000000000000000000000000000;
    0B   :   00000000000000000000000000000000;
    0C   :   00000000000000000000000000000000;
    0D   :   00000000000000000000000000000000;
```

Single cycle implementation
Instruction Memory

```
CONTENT BEGIN
    00   :   00100000000010010000000100010010;--load phase addi $t1, $zero, 0x112, so that $t1 = 0x112
    01   :   00100000000010100000000000001010;--load phase addi $t2. $zero, 0xA, so that $t2 = 0xA
    02   :   00100000000010110000000000001111;--load phase addi $t3, $zero, 0xF, so that $t3 = 0xF
    03   :   00000000000000000000000000000000;--stall
    04   :   00000000000000000000000000000000;--stall
    05   :   00000000000000000000000000000000;--stall
    06   :   00010001001010100000000000000011;--beq $t1, $t2, Equal
    07   :   00000001001010100100100000100000;--add $t1, $t1, $t2 (result 0x11C)
    08   :   10101101010010110000000001100100;--sw $t3, 100($t2)
    09   :   00000001001010100100100000100101;--or $t1, $t1, $t2 (result 0x11A)
    0A   :   00000000000000000000000000000000;--equal
    0B   :   00000000000000000000000000000000;
    0C   :   00000000000000000000000000000000;
    0D   :   00000000000000000000000000000000;
```

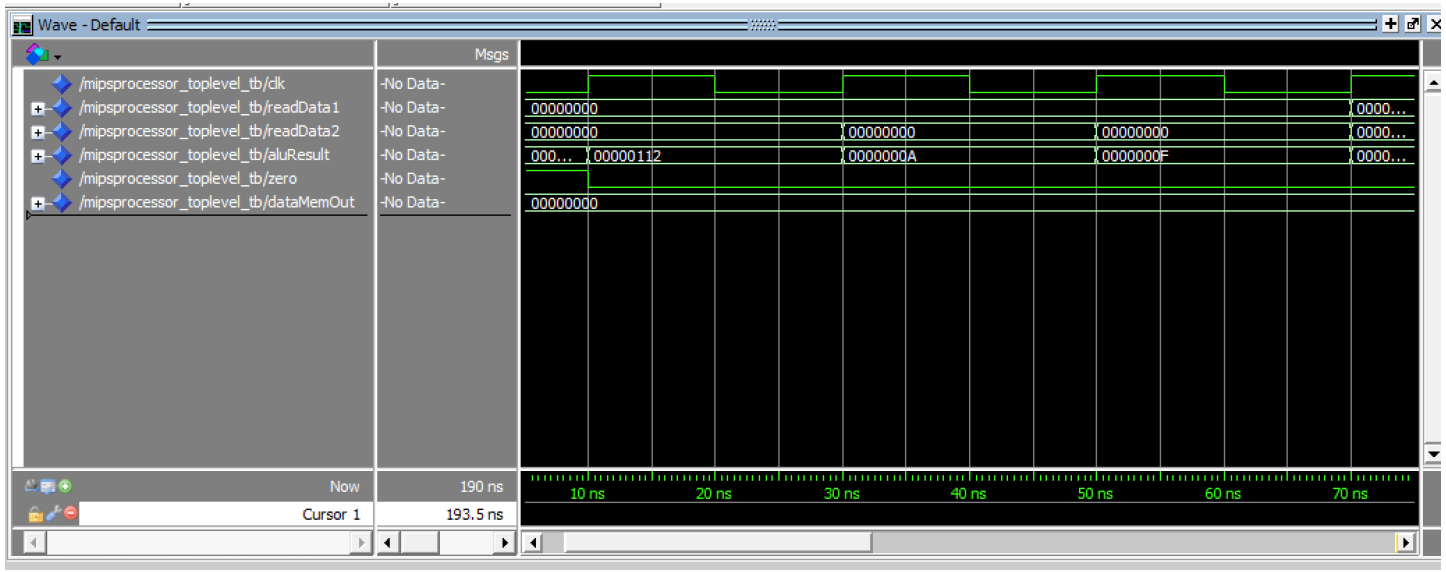
Pipeline implementation
Instruction memory

Data memory implantation
Memory initialization file

```
CONTENT BEGIN
    00   :   00000000000000000000000000000000;
    01   :   00000000000000000000000000000000;
    02   :   00000000000000000000000000000000;
    03   :   00000000000000000000000000000000;
    04   :   00000000000000000000000000000000;
    05   :   00000000000000000000000000000000;
    06   :   00000000000000000000000000000000;
    07   :   00000000000000000000000000000000;
    08   :   00000000000000000000000000000000;
    09   :   00000000000000000000000000000000;
    0A   :   00000000000000000000000000000000;
    0B   :   00000000000000000000000000000000;
    0C   :   00000000000000000000000000000000;
    0D   :   00000000000000000000000000000000;
    0E   :   00000000000000000000000000000000;
    0F   :   00000000000000000000000000000000;
    10   :   00000000000000000000000000000000;
    11   :   00000000000000000000000000000000;
    12   :   00000000000000000000000000000000;
    13   :   00000000000000000000000000000000;
    14   :   00000000000000000000000000000000;
    15   :   00000000000000000000000000000000;
    16   :   00000000000000000000000000000000;
    17   :   00000000000000000000000000000000;
    18   :   00000000000000000000000000000000;
    19   :   00000000000000000000000000000000;
    1A   :   00000000000000000000000000000000;
    1B   :   00000000000000000000000000000000;
    1C   :   00000000000000000000000000000000;
    1D   :   00000000000000000000000000000000;
```

Single cycle successful implementation

Register load phase



Program Execution Phase

beq: 0x112 – 0xA = 0x108 != 0 (branch not taken) ✓

add: 0x112 + 0xA = 0x11C ✓

sw: dataMemOut updated next cycle, because $t3 = 0xF

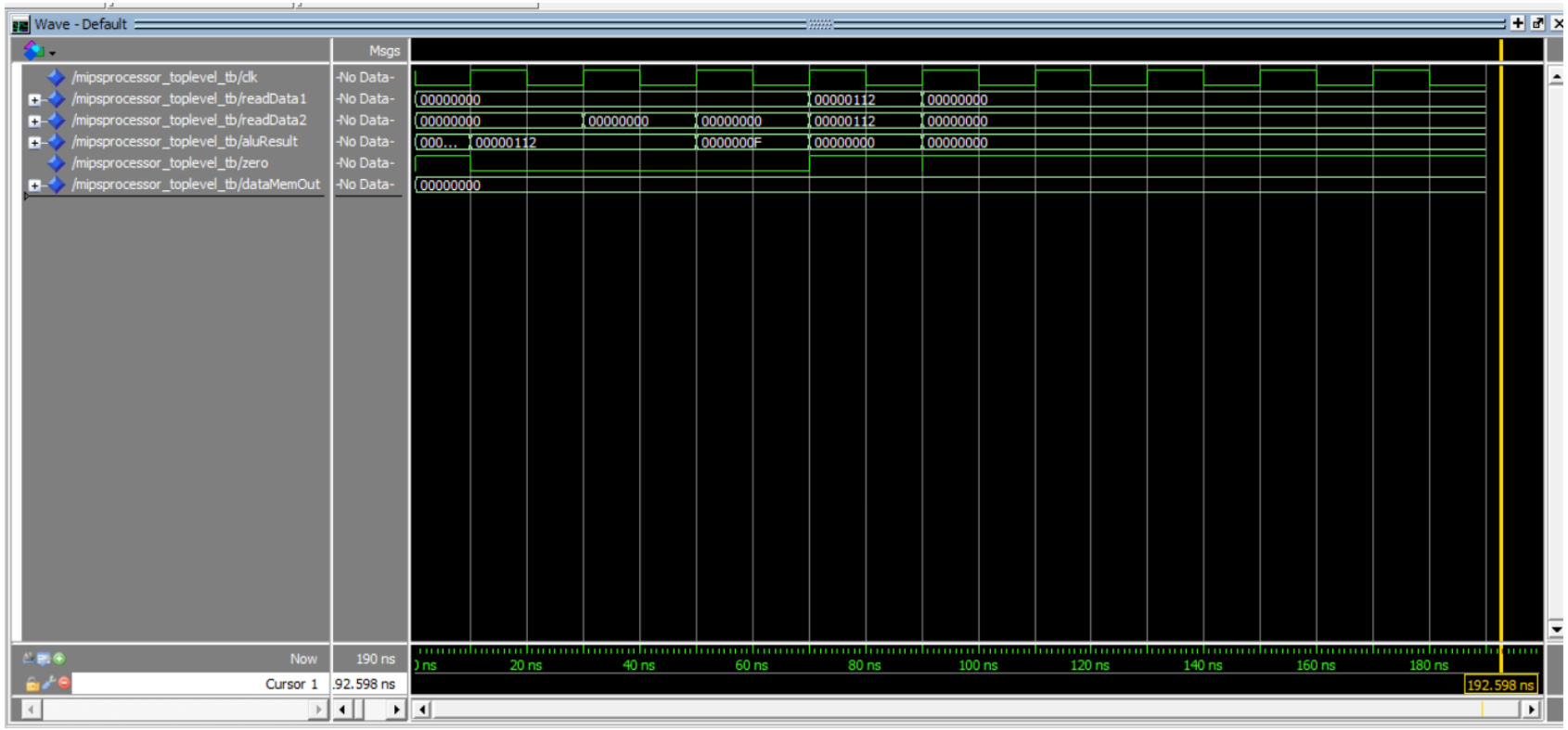
or: 0x11C OR 0xA = 0x11E ✓

Nathan Gardner

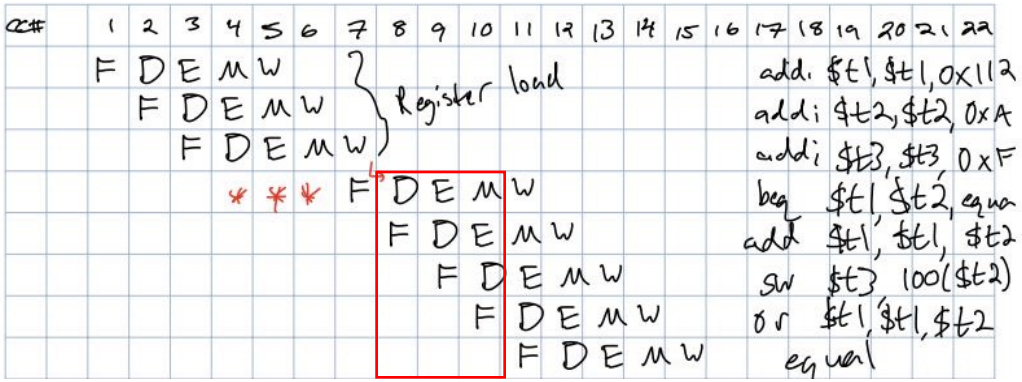
Tennessee TECH

Single cycle successful implementation (beq taken)

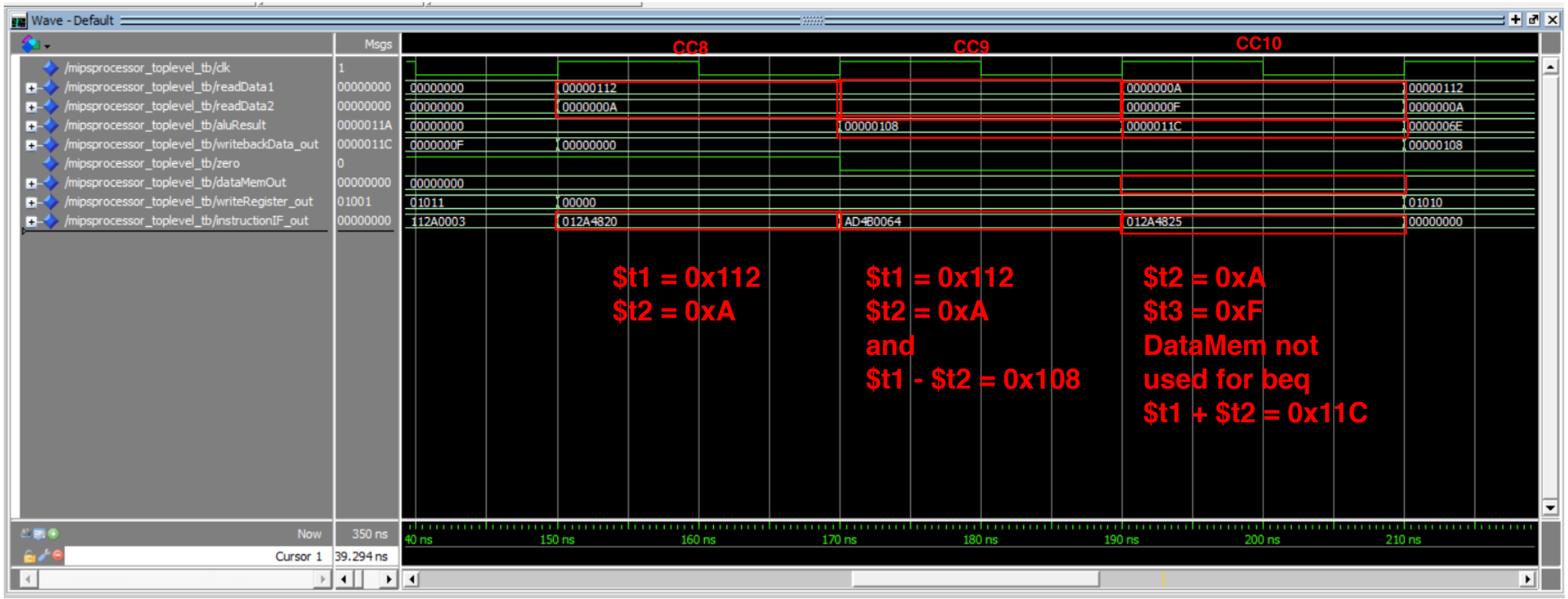
```
CONTENT BEGIN
    00   :   00100000000010010000000100010010;--load phase addi $t1, $zero, 0x112, so that $t1 = 0x112
    01   :   00100000000010100000000100010010;--load phase addi $t2, $zero, 0xA, so that $t2 = 0x112
    02   :   00100000000010110000000000001111;--load phase addi $t3, $zero, 0xF, so that $t3 = 0xF
    03   :   00010001001010100000000000000011;--beq $t1, $t2, Equal
    04   :   00000001001010100100100000100000;--add $t1, $t1, $t2
    05   :   10101101010010110000000001100100;--sw $t3, 100($t2)
    06   :   00000001001010100100100000100101;--or $t1, $t1, $t2
    07   :   00000000000000000000000000000000;--equal
    08   :   00000000000000000000000000000000;
    09   :   00000000000000000000000000000000;
    0A   :   00000000000000000000000000000000;
    0B   :   00000000000000000000000000000000;
    0C   :   00000000000000000000000000000000;
    0D   :   00000000000000000000000000000000;
```



Pipeline successful implementation



Show successful implementation of pipeline with clock cycles 8,9, and 10.



Nathan Gardner

Tennessee TECH