Project 2

ECE2500 CRN:82943

Jacob Abel

November 27, 2018

Summary

This project involved the development of a MIPS Assembly function to compute the greatest common denominator of the inputs and the implementation of the required instructions in Verilog to run said function. The iterative C function that the Assembly is based on is included in Appendix A.

Assembly Design

The function was relatively simple to implement. Each standard form (if, for, while, do-while) in the C function was delimited by labels in the assembly source. At this point the labels were filled in with the corresponding assembly to each standard form. Minor mistakes were made such as writing a bne instead of a beq but these were easily resolved by analysing the failing loops. The function ended up requiring the following functions: beq, bne, andi, or, addi, srl, sll, slt, & sub. These are implemented in the next section. The main function for the QTSpim simulation was a series of straightforward load immediates, moves, and jump-and-links. As such the design was trivial. The simulation results of the QTSpim run are in Appendices B and C.

Verilog Design

The modifications to the existing Verilog were also fairly trivial. Within `mips-control.v` each instruction was added to the `casex` based on their corresponding values from the MIPS Reference Sheets. The procedural blocks for each statement was adopted from their corresponding instruction types. The R-types adopted from the sll instruction and the I-types were adopted from the addi instruction. The ALU codes were determined by reading `mips-alu.v`. Branching was implemented by adding a 2x1 mux and 2 dataflow statements to `mips.v`. The dataflow statements generated the branch PC address and the enable bit for the mux. The mux rerouted the input to the PC reg from the incrementer to the branch address when the enable bit was high. The branch instructions were adopted from the R-type's procedural block and the `branch_out` wires were enabled. The `jump_out` was re-purposed as an enable for inverting the branch's zero check. Luckily no debugging was required as the project compiled and simulated correctly on the first attempt. The simulation results are in Appendix D.

Appendix A: GCD C Source

```c
unsigned int gcd(unsigned int u, unsigned int v)
{
    int shift;
    // GCD(0,v) == v; GCD(u,0) == u, GCD(0,0) == 0
    if (u == 0) return v;
    if (v == 0) return u;
    /*
     * Let shift := lg K, where K is the greatest power of 2
     * dividing both u and v.
     */
    for (shift = 0; ((u | v) & 1) == 0; ++shift) {
        u >>= 1;
        v >>= 1;
    }
    while ((u & 1) == 0) {
        u >>= 1;
    }
    // From here on, u is always odd.
    do {
        // remove all factors of 2 in v -- they are not common
        //   note: v is not zero, so while will terminate
        while ((v & 1) == 0) { /* Loop X */
            v >>= 1;
        }
        /*
         * Now u and v are both odd. Swap if necessary so u <= v,
         * then set v = v - u (which is even). For bignums, the
         * swapping is just pointer movement, and the subtraction
         * can be done in-place.
         */
        if (u > v) {
            unsigned int t = v; v = u; u = t; // Swap u and v.
        }
        v = v - u; // Here v >= u.
    } while (v != 0);
    // restore common factors of 2
    return u << shift;
}
```

Iterative C Greatest Common Denominator Function (courtesy Wikipedia)

Appendix B: BinaryGCD in QTSpim prior to running

```
PC        = 0
EPC       = 0
Cause     = 0
BadVAddr  = 0
Status    = 3000ff10

HI        = 0
LO        = 0

R0  [r0]  = 0
R1  [at]  = 0
R2  [v0]  = 0
R3  [v1]  = 0
R4  [a0]  = 1
R5  [a1]  = 7ffff00c
R6  [a2]  = 7ffff014
R7  [a3]  = 0
R8  [t0]  = 0
R9  [t1]  = 0
R10 [t2]  = 0
R11 [t3]  = 0
R12 [t4]  = 0
R13 [t5]  = 0
R14 [t6]  = 0
R15 [t7]  = 0
R16 [s0]  = 0
R17 [s1]  = 0
R18 [s2]  = 0
R19 [s3]  = 0
R20 [s4]  = 0
R21 [s5]  = 0
R22 [s6]  = 0
R23 [s7]  = 0
R24 [t8]  = 0
R25 [t9]  = 0
R26 [k0]  = 0
R27 [k1]  = 0
R28 [gp]  = 10008000
R29 [sp]  = 7ffff008
R30 [s8]  = 0
R31 [ra]  = 0
```

```
[004000b4] 0c100038  jal 0x004000e0 [binaryGCD]; 35: jal binaryGCD
[004000b8] 0002b021  addu $22, $0, $2          ; 36: move $s6, $v0
[004000bc] 0010f821  addu $31, $0, $16         ; 37: move $ra, $s0
[004000c0] 34040000  ori $4, $0, 0             ; 38: li $a0, 0
[004000c4] 34053039  ori $5, $0, 12345         ; 39: li $a1, 12345
[004000c8] 001f8021  addu $16, $0, $31         ; 40: move $s0, $ra
[004000cc] 0c100038  jal 0x004000e0 [binaryGCD]; 41: jal binaryGCD
[004000d0] 0002b821  addu $23, $0, $2          ; 42: move $s7, $v0
[004000d4] 0010f821  addu $31, $0, $16         ; 43: move $ra, $s0
[004000d8] 3402000a  ori $2, $0, 10            ; 44: ori $v0, $zero, 10
[004000dc] 0000000c  syscall                   ; 45: syscall
[004000e0] 20a20000  addi $2, $5, 0            ; 49: addi $v0, $a1, 0
[004000e4] 10800021  beq $4, $0, 132 [bGCD_e-0x004000e4]
[004000e8] 20820000  addi $2, $4, 0            ; 51: addi $v0, $a0, 0
[004000ec] 10a0001f  beq $5, $0, 124 [bGCD_e-0x004000ec]
[004000f0] 20880000  addi $8, $4, 0            ; 54: addi $t0, $a0, 0
[004000f4] 20a90000  addi $9, $5, 0            ; 55: addi $t1, $a1, 0
[004000f8] 200a0000  addi $10, $0, 0           ; 56: addi $t2, $zero, 0
[004000fc] 01095825  or $11, $8, $9            ; 59: or $t3, $t0, $t1
[00400100] 316b0001  andi $11, $11, 1          ; 60: andi $t3, $t3, 1
[00400104] 15600005  bne $11, $0, 20 [bGCD_for_e-0x00400104]
[00400108] 00084042  srl $8, $8, 1             ; 62: srl $t0, $t0, 1
[0040010c] 00094842  srl $9, $9, 1             ; 63: srl $t1, $t1, 1
[00400110] 214a0001  addi $10, $10, 1          ; 64: addi $t2, $t2, 1
[00400114] 1000fffa  beq $0, $0, -24 [bGCD_for-0x00400114]
[00400118] 310b0001  andi $11, $8, 1           ; 69: andi $t3, $t0, 1
[0040011c] 15000003  bne $8, $0, 12 [bGCD_while0_e-0x0040011c]
[00400120] 00084042  srl $8, $8, 1             ; 71: srl $t0, $t0, 1
[00400124] 1000fffd  beq $0, $0, -12 [bGCD_while0-0x00400124]
[00400128] 312b0001  andi $11, $9, 1           ; 77: andi $t3, $t1, 1
[0040012c] 15600003  bne $11, $0, 12 [bGCD_while1_e-0x0040012c]
[00400130] 00094842  srl $9, $9, 1             ; 79: srl $t1, $t1, 1
[00400134] 1000fffd  beq $0, $0, -12 [bGCD_while1-0x00400134]
[00400138] 0128582a  slt $11, $9, $8           ; 83: slt $t3, $t1, $t0
[0040013c] 11600004  beq $11, $0, 16 [bGCD_if_e-0x0040013c]
[00400140] 210c0000  addi $12, $8, 0           ; 86: addi $t4, $t0, 0
[00400144] 21280000  addi $8, $9, 0            ; 87: addi $t0, $t1, 0
[00400148] 21890000  addi $9, $12, 0           ; 88: addi $t1, $t4, 0
[0040014c] 01284822  sub $9, $9, $8            ; 90: sub $t1, $t1, $t0
[00400150] 1520fff6  bne $9, $0, -40 [bGCD_do-0x00400150]
[00400154] 11400004  beq $10, $0, 16 [bGCD_sll_e-0x00400154]
[00400158] 00084040  sll $8, $8, 1             ; 95: sll $t0, $t0, 1
[0040015c] 214affff  addi $10, $10, -1         ; 96: addi $t2, $t2, -1
[00400160] 1000fffd  beq $0, $0, -12 [bGCD_sll-0x00400160]
[00400164] 21020000  addi $2, $8, 0            ; 99: addi $v0, $t0, 0
[00400168] 03e00008  jr $31                    ; 101: jr $ra
```

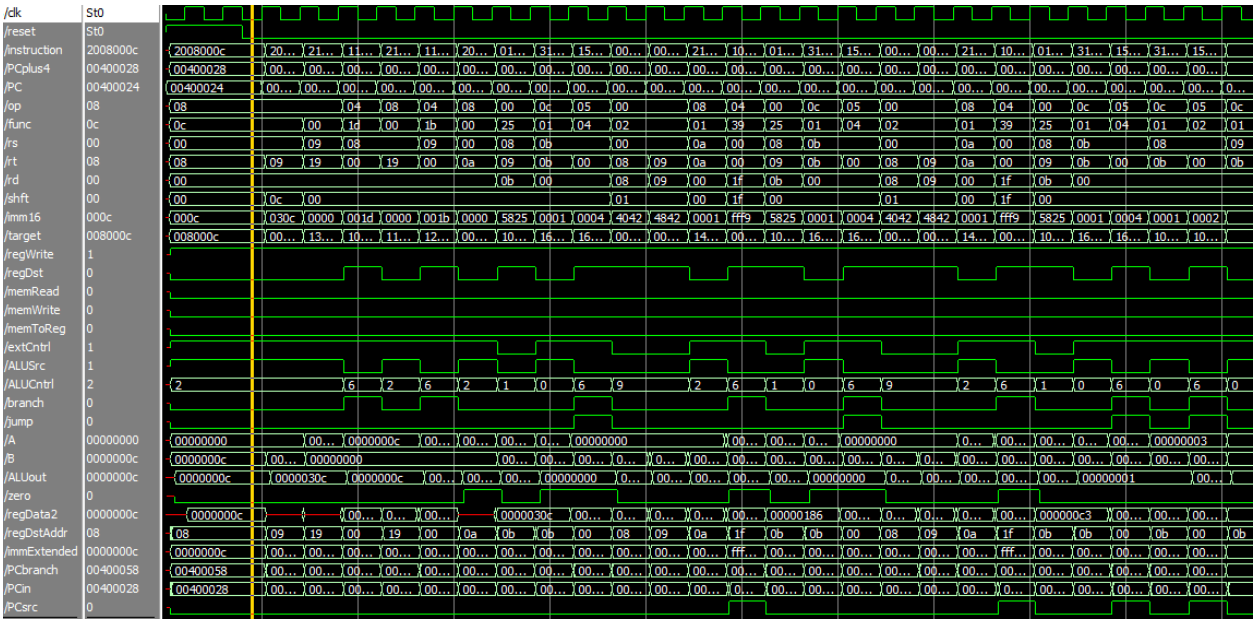
Appendix C: BinaryGCD in QTSpim after running

```
PC        = 4000dc
EPC       = 0
Cause     = 0
BadVAddr  = 0
Status    = 3000ff10

HI        = 0
LO        = 0

R0  [r0] = 0
R1  [at] = 90000
R2  [v0] = a
R3  [v1] = 0
R4  [a0] = 0
R5  [a1] = 3039
R6  [a2] = 7ffff014
R7  [a3] = 0
R8  [t0] = 20000
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 8
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 400018
R17 [s1] = c
R18 [s2] = c
R19 [s3] = 800
R20 [s4] = 1
R21 [s5] = c4e0
R22 [s6] = 20000
R23 [s7] = 3039
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff008
R30 [s8] = 0
R31 [ra] = 400018
```

```
[004000b4] 0c100038  jal 0x004000e0 [binaryGCD]; 35: jal binaryGCD
[004000b8] 0002b021  addu $22, $0, $2          ; 36: move $s6, $v0
[004000bc] 0010f821  addu $31, $0, $16         ; 37: move $ra, $s0
[004000c0] 34040000  ori $4, $0, 0             ; 38: li $a0, 0
[004000c4] 34053039  ori $5, $0, 12345         ; 39: li $a1, 12345
[004000c8] 001f8021  addu $16, $0, $31         ; 40: move $s0, $ra
[004000cc] 0c100038  jal 0x004000e0 [binaryGCD]; 41: jal binaryGCD
[004000d0] 0002b821  addu $23, $0, $2          ; 42: move $s7, $v0
[004000d4] 0010f821  addu $31, $0, $16         ; 43: move $ra, $s0
[004000d8] 3402000a  ori $2, $0, 10            ; 44: ori $v0, $zero, 10
[004000dc] 0000000c  syscall                   ; 45: syscall
[004000e0] 20a20000  addi $2, $5, 0            ; 49: addi $v0, $a1, 0
[004000e4] 10800021  beq $4, $0, 132 [bGCD_e-0x004000e4]
[004000e8] 20820000  addi $2, $4, 0            ; 51: addi $v0, $a0, 0
[004000ec] 10a0001f  beq $5, $0, 124 [bGCD_e-0x004000ec]
[004000f0] 20880000  addi $8, $4, 0            ; 54: addi $t0, $a0, 0
[004000f4] 20a90000  addi $9, $5, 0            ; 55: addi $t1, $a1, 0
[004000f8] 200a0000  addi $10, $0, 0           ; 56: addi $t2, $zero, 0
[004000fc] 01095825  or $11, $8, $9            ; 59: or $t3, $t0, $t1
[00400100] 316b0001  andi $11, $11, 1          ; 60: andi $t3, $t3, 1
[00400104] 15600005  bne $11, $0, 20 [bGCD_for_e-0x00400104]
[00400108] 00084042  srl $8, $8, 1             ; 62: srl $t0, $t0, 1
[0040010c] 00094842  srl $9, $9, 1             ; 63: srl $t1, $t1, 1
[00400110] 214a0001  addi $10, $10, 1          ; 64: addi $t2, $t2, 1
[00400114] 1000fffa  beq $0, $0, -24 [bGCD_for-0x00400114]
[00400118] 310b0001  andi $11, $8, 1           ; 69: andi $t3, $t0, 1
[0040011c] 15000003  bne $8, $0, 12 [bGCD_while0_e-0x0040011c]
[00400120] 00084042  srl $8, $8, 1             ; 71: srl $t0, $t0, 1
[00400124] 1000fffd  beq $0, $0, -12 [bGCD_while0-0x00400124]
[00400128] 312b0001  andi $11, $9, 1           ; 77: andi $t3, $t1, 1
[0040012c] 15600003  bne $11, $0, 12 [bGCD_while1_e-0x0040012c]
[00400130] 00094842  srl $9, $9, 1             ; 79: srl $t1, $t1, 1
[00400134] 1000fffd  beq $0, $0, -12 [bGCD_while1-0x00400134]
[00400138] 0128582a  slt $11, $9, $8           ; 83: slt $t3, $t1, $t0
[0040013c] 11600004  beq $11, $0, 16 [bGCD_if_e-0x0040013c]
[00400140] 210c0000  addi $12, $8, 0           ; 86: addi $t4, $t0, 0
[00400144] 21280000  addi $8, $9, 0            ; 87: addi $t0, $t1, 0
[00400148] 21890000  addi $9, $12, 0           ; 88: addi $t1, $t4, 0
[0040014c] 01284822  sub $9, $9, $8            ; 90: sub $t1, $t1, $t0
[00400150] 1520fff6  bne $9, $0, -40 [bGCD_do-0x00400150]
[00400154] 11400004  beq $10, $0, 16 [bGCD_sll_e-0x00400154]
[00400158] 00084040  sll $8, $8, 1             ; 95: sll $t0, $t0, 1
[0040015c] 214affff  addi $10, $10, -1         ; 96: addi $t2, $t2, -1
[00400160] 1000fffd  beq $0, $0, -12 [bGCD_sll-0x00400160]
[00400164] 21020000  addi $2, $8, 0            ; 99: addi $v0, $t0, 0
[00400168] 03e00008  jr $31                    ; 101: jr $ra
```

Appendix D: Modelsim waveform of BinaryGCD(12, 780)

At Initialisation



Full Run (Including Final Results)