CSE3666 HW4 solutions

1. State machine.

The truth table is as follows.

state[1]	state[0]	b	next_state[1]	next_state[0]	z
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	0	1	0

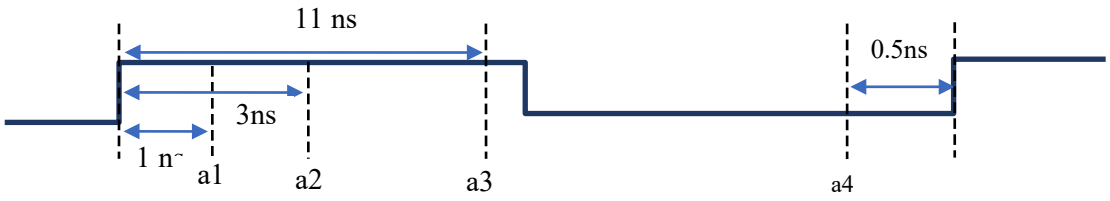
It is trivial to write down the logic expression for each signal.

The output of the program is as follows.

```
state b | ns z v
  0   1 | 1  0 1
  1   1 | 0  1 3
  0   1 | 1  0 7
  1   0 | 2  0 14
  2   0 | 1  0 28
  1   0 | 2  0 56
  2   1 | 2  0 113
  2   0 | 1  0 226
  1   1 | 0  1 453
  0   1 | 1  0 907
```

2. Timing.

a)



Time relative to the beginning of a cycle.

a1. Reg-Ready. 1ns

a2. Control-Ready. 1 + 2 = 3ns. 2 is the delay of the combinational circuit in the control module.

a3. Adder-Ready. 1 + 10 = 11ns. 10 is the delay of the adder.

a4. Deadline. The input to registers must be ready at least 0.5ns before the rising edge.

The combinational circuit has their output ready 11 ns into a cycle. The setup time is 0.5 ns.

b)

The minimum cycle time is: 11 + 0.5 = 11.5 ns.

c)

The highest clock rate is: 1 / 11.5 ns = 0.08696 GHz = 87.0 MHz

d)

If we use the same kind of registers, the max clock rate we can achieve is 1/(1ns + 0.5ns) = 0.6667 GHz = 666.7 MHz. In this case, there is no combinational circuit (or no delay other than the register itself).

3. Multiplier.

27 * 17 = 459

```
0     0000011011    10001  0000000000
1     0000110110    01000  0000011011
2     0001101100    00100  0000011011
3     0011011000    00010  0000011011
4     0110110000    00001  0000011011
5     1101100000    00000  0111001011
```

If we consider the bits are two's complement number, 0b11011 is -5 and 0b10001 is -15. Their product is 75, which is 0b00010_01011. The lowest five bits are the same as the lowest five bits in the product register.

4. Coding with div.

```
# char * uint2decstr(char s[], unsigned int v)
uint2decstr:
      # save ra and a1 (v) on the stack
      addi  sp, sp, -8
      sw    ra, 4(sp)
      sw    a1, 0(sp)

      # if (v >= 10)
      addi  t1, zero, 10
      bltu  a1, t1, skip

      # if branch
      #     s = uint2decstr(s, v / 10);
      # a0 is already set
      divu  a1, a1, t1
      jal   ra, uint2decstr

skip:
      # r = v % 10
      lw    a1, 0(sp)        # restore v
      addi  t1, zero, 10     # t1 may have been changed
      remu  a1, a1, t1

      # s is in a0, and r in a1
      # s[0] = '0' + r;
      # s[1] = 0;
      addi  a1, a1, '0'
      sb    a1, 0(a0)
      sb    zero, 1(a0)

      # return s + 1;
      addi  a0, a0, 1

      # no need to restore register a1
      lw    ra, 4(sp)
      addi  sp, sp, 8
      jalr  x0, ra, 0
```