Opcode = 0000 -> ACC <= C

T0 : func = C , acc_ld = 1

Opcode = 0001 -> ACC <= D

T0 : func = D , acc_ld = 1

Opcode = 0010 -> ACC <= C + D

T0 : func = add , acc_ld = 1

Opcode = 0011 -> ACC <= B and C

T0 : sel = 00 , d_ld = 1

T1 : func = and , acc_ld = 1

Opcode = 0100 -> ACC <= B xor C

T0 : sel = 00 , d_ld = 1

T1 : func = xor , acc_ld = 1

Opcode = 0101 -> ACC <= mem[A]

T0 : sel = 10 , c_ld = 1

T1 : func = C , acc_ld = 1

Opcode = 0110 -> mem[A] <= ACC

T0 : sel = 01 , wr = 1

Opcode = 0111 -> ACC <= mem[B] + mem[C]

T0 : sel = 00 , a_ld = 1

T1 : sel = 10 , d_ld = 1

T2 : func = C , acc_ld = 1

T3 : sel = 01 , a_ld = 1

T4 : sel = 10 , c_ld = 1

T5 : func = add , acc_ld = 1

Opcode = 1000 -> ACC <= mem[B] – mem[C]

T0 : func = C , acc_ld = 1

T1 : sel = 01 , a_ld = 1

T2 : sel = 10 , d_ld = 1

T3 : sel = 00 , a_ld = 1

T4 : sel = 10 , c_ld = 1

T5 : func = sub , acc_ld = 1

Opcode = 1001 -> mem[A] <= B xor C

T0 : sel = 00 , d_ld = 1

T1 : func = xor , acc_ld = 1

T2 : sel = 01 , wr = 1

Opcode = 1010 -> mem[A] <= B << C[4:0]

T0 : sel = 00 , d_ld = 1

T1 : func = shift , acc_ld = 1

T2 : sel = 01 , wr = 1