



# **CprE 281: Digital Logic**

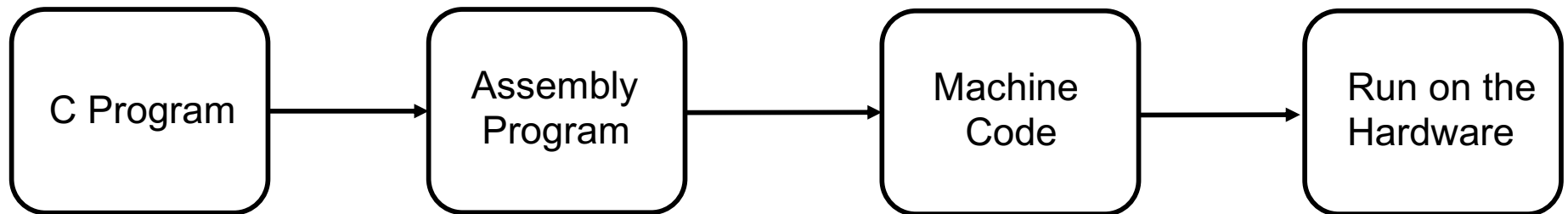
**Instructor: Alexander Stoytchev**

**<http://www.ece.iastate.edu/~alexs/classes/>**

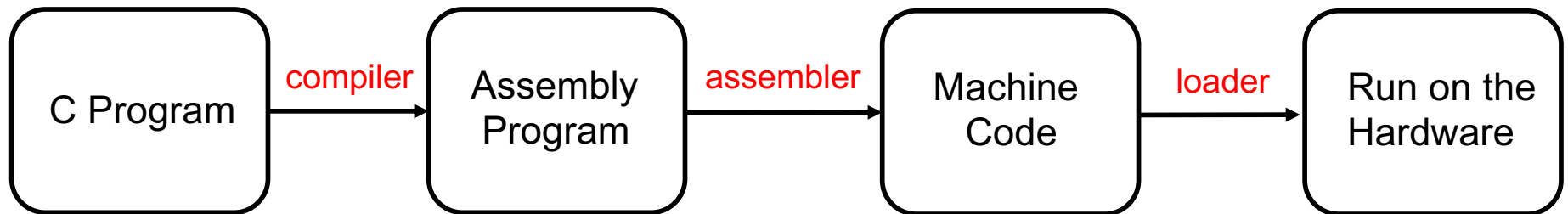
# The Intersection Between Hardware and Software

*CprE 281: Digital Logic  
Iowa State University, Ames, IA  
Copyright © Alexander Stoytchev*

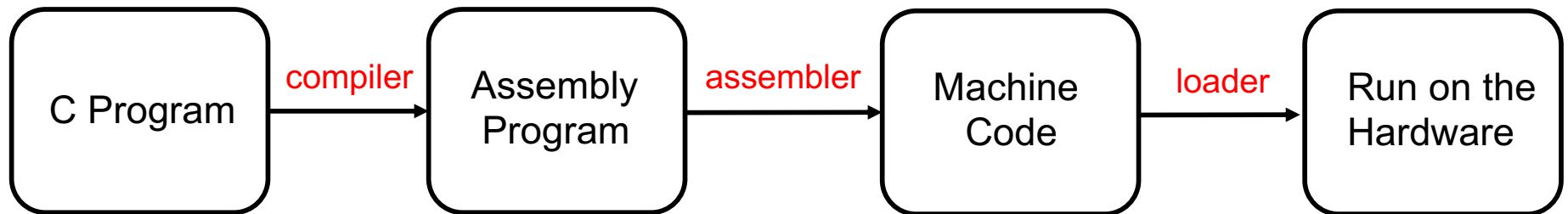
# Writing and Running a Program



# Writing and Running a Program

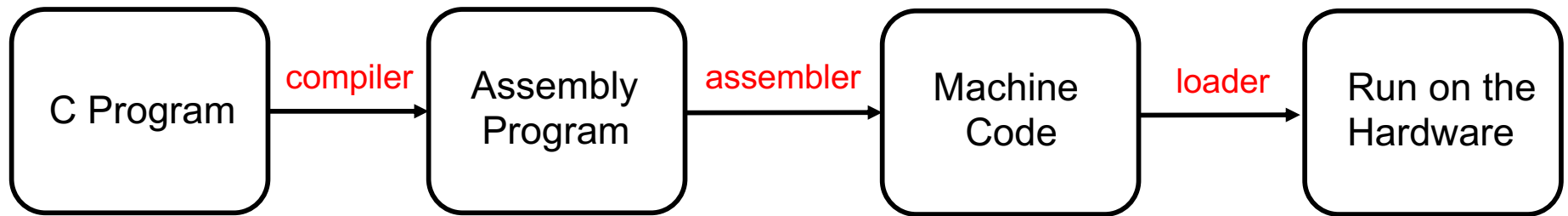


# Writing and Running a Program



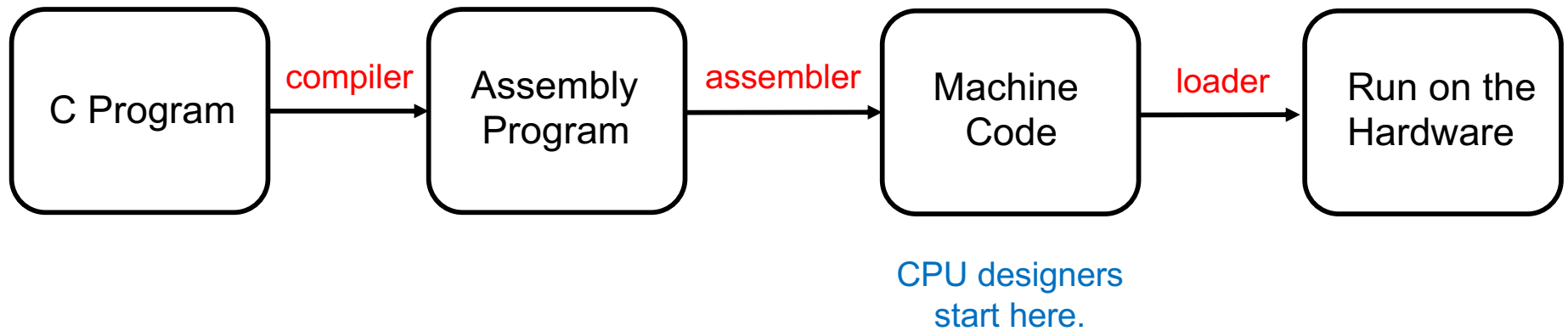
The programmer only writes this in a text editor.

# Writing and Running a Program



Nerds skip the first step and start here.

# Writing and Running a Program



**i281 Example:**  
**Add the numbers from 1 to 5**



**i281 Example:**  
**Add the numbers from 1 to 5**

**C Language v.s. Assembly Language**

# C Version

```
// C Version
//
// Add the numbers from 1 to 5 using a for loop.

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++)
        sum+=i;

    // printf("%d\n", sum);
}
```

# i281 Assembly Version

**.data**

```
N          BYTE    5
i          BYTE    ?
sum        BYTE    ?
```

**.code**

```
          LOADI   B, 0          ; sum=0
          LOADI   A, 1          ; i=1
          LOAD    D, [N]        ; register_D=N
Loop:     CMP     A, D          ; i<=N ?
          BRG     End          ; exit if i>N
Add:      ADD     B, A          ; sum+=i
          ADDI    A, 1          ; i++
          JUMP    Loop         ; next iteration
End:      STORE   [sum], B      ; update the memory for sum
```

**; Register allocation:**

**; A: i**

**; B: sum**

**; C: <not used>**

**; D: N**

# i281 Assembly Version

**.data**

```
N      BYTE    5
i      BYTE    ?
sum    BYTE    ?
```

**.code**

```
        LOADI   B, 0      ; sum=0
        LOADI   A, 1      ; i=1
        LOAD    D, [N]    ; register_D=N
Loop:   CMP     A, D      ; i<=N ?
        BRG     End      ; exit if i>N
Add:    ADD     B, A      ; sum+=i
        ADDI   A, 1      ; i++
        JUMP   Loop      ; next iteration
End:    STORE   [sum], B  ; update the memory for sum
```

**; Register allocation:**

**; A: i**

**; B: sum**

**; C: <not used>**

**; D: N**

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N          BYTE    5
i          BYTE    ?
sum        BYTE    ?

.code

        LOADI    B, 0        ; sum=0
        LOADI    A, 1        ; i=1
        LOAD     D, [N]      ; register_D=N
Loop:   CMP      A, D        ; i<=N ?
        BRG      End        ; exit if i>N
Add:    ADD      B, A        ; sum+=i
        ADDI    A, 1        ; i++
        JUMP    Loop        ; next iteration
End:    STORE    [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N          BYTE    5
i          BYTE    ?
sum        BYTE    ?

.code

        LOADI   B, 0          ; sum=0
        LOADI   A, 1          ; i=1
        LOAD    D, [N]        ; register_D=N
Loop:    CMP     A, D          ; i<=N ?
        BRG     End          ; exit if i>N
Add:     ADD    B, A          ; sum+=i
        ADDI   A, 1          ; i++
        JUMP   Loop          ; next iteration
End:     STORE  [sum], B      ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```



# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI  B, 0        ; sum=0
        LOADI  A, 1        ; i=1
        LOAD   D, [N]      ; register_D=N
Loop:   CMP    A, D        ; i<=N ?
        BRG    End        ; exit if i>N
Add:    ADD    B, A        ; sum+=i
        ADDI  A, 1        ; i++
        JUMP   Loop       ; next iteration
End:    STORE [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

**i=1**

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI  A, 1        ; i=1
        LOAD   D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG    End        ; exit if i>N
Add:    ADD    B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop       ; next iteration
End:    STORE  [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

This has no analog in the C version,  
which is written in a high-level language.

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

Load the value of N into register D.

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]     ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD    B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop       ; next iteration
End:    STORE  [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

**i=2**

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```



# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE  [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

**i=3**

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE  [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD    B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop       ; next iteration
End:    STORE  [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

**i=4**

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:    CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:     ADD    B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:     STORE  [sum], B    ; write B to sum
```



# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop       ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE  [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

**i=5**

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE  [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]     ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

**i=6**

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```



# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD    B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE [sum], B    ; write B to sum
```

**i281 Example:  
Add the numbers from 1 to 5**

**Assembly Language v.s. Machine Language**

# i281 Assembly Code

**.data**

```
N          BYTE    5
i          BYTE    ?
sum       BYTE    ?
```

**.code**

```
          LOADI  B, 0          ; sum=0
          LOADI  A, 1          ; i=1
          LOAD   D, [N]        ; register_D=N
Loop:    CMP    A, D          ; i<=N ?
          BRG    End          ; exit if i>N
Add:    ADD    B, A          ; sum+=i
          ADDI   A, 1          ; i++
          JUMP   Loop         ; next iteration
End:    STORE  [sum], B     ; update the memory for sum
```

# i281 Assembly Code

**.data**

**N            BYTE     5**

**i            BYTE     ?**

**sum          BYTE     ?**

**.code**

**LOADI    B, 0**

**LOADI    A, 1**

**LOAD     D, [N]**

**Loop:     CMP      A, D**

**BRG      End**

**Add:     ADD      B, A**

**ADDI     A, 1**

**JUMP     Loop**

**End:      STORE    [sum], B**

# Mapping Assembly to Machine Code

**.data**

```
N      BYTE  5
i      BYTE  ?
sum    BYTE  ?
```

**Data Memory:**

```
00000101
00000000
00000000
```

**.code**

```
      LOADI  B, 0
      LOADI  A, 1
      LOAD   D, [N]
Loop:  CMP    A, D
      BRG    End
Add:   ADD   B, A
      ADDI  A, 1
      JUMP  Loop
End:   STORE [sum], B
```

**Code Memory:**

```
0011010000000000
0011000000000001
1000110000000000
1101001100000000
1111001000000011
0100010000000000
0101000000000001
1110000011111011
1010010000000010
```

Assembly Language

Machine Language

# Mapping Assembly to Machine Code

**.data**

```
N      BYTE  5
i      BYTE  ?
sum    BYTE  ?
```

**Data Memory:**

```
0000 0101
0000 0000
0000 0000
```

**.code**

```
      LOADI  B, 0
      LOADI  A, 1
      LOAD   D, [N]
Loop:  CMP    A, D
      BRG    End
Add:   ADD   B, A
      ADDI  A, 1
      JUMP  Loop
End:   STORE [sum], B
```

**Code Memory:**

```
0011 0100 0000 0000
0011 0000 0000 0001
1000 1100 0000 0000
1101 0011 0000 0000
1111 0010 0000 0011
0100 0100 0000 0000
0101 0000 0000 0001
1110 0000 1111 1011
1010 0100 0000 0010
```

Assembly Language

Machine Language  
in Binary

# Mapping Assembly to Machine Code

**.data**

**N**        **BYTE**    **5**  
**i**        **BYTE**    **?**  
**sum**     **BYTE**    **?**

**Data Memory:**

**0**    **5**  
**0**    **0**  
**0**    **0**

**.code**

**LOADI**   **B, 0**  
          **LOADI**   **A, 1**  
          **LOAD**    **D, [N]**  
**Loop:**    **CMP**     **A, D**  
          **BRG**     **End**  
**Add:**    **ADD**     **B, A**  
          **ADDI**    **A, 1**  
          **JUMP**    **Loop**  
**End:**     **STORE**   **[sum], B**

**Code Memory:**

**3**    **4**    **0**    **0**  
**3**    **0**    **0**    **1**  
**8**    **C**    **0**    **0**  
**D**    **3**    **0**    **0**  
**F**    **2**    **0**    **3**  
**4**    **4**    **0**    **0**  
**5**    **0**    **0**    **1**  
**E**    **0**    **F**    **B**  
**A**    **4**    **0**    **2**

Assembly Language

Machine Language  
in Binary

# Mapping Assembly to Machine Code

**.data**

**N**        **BYTE**    **5**  
**i**        **BYTE**    **?**  
**sum**     **BYTE**    **?**

**Data Memory:**

**05**  
**00**  
**00**

**.code**

**LOADI**  **B, 0**  
          **LOADI**  **A, 1**  
          **LOAD**   **D, [N]**  
**Loop:**    **CMP**   **A, D**  
          **BRG**   **End**  
**Add:**    **ADD**   **B, A**  
          **ADDI**  **A, 1**  
          **JUMP**  **Loop**  
**End:**     **STORE**  **[sum], B**

**Code Memory:**

**34 00**  
**30 01**  
**8C 00**  
**D3 00**  
**F2 03**  
**44 00**  
**50 01**  
**E0 FB**  
**A4 02**

Assembly Language

Machine Language  
in Hexadecimal



**i281 Example:  
Add the numbers from 1 to 5**

**Bit Mapping for OPCODEs**

# Mapping Assembly to Machine Code

**.data**

```
N      BYTE  5
i      BYTE  ?
sum    BYTE  ?
```

**Data Memory:**

```
00000101
00000000
00000000
```

**.code**

```
      LOADI  B, 0
      LOADI  A, 1
      LOAD   D, [N]
Loop:  CMP    A, D
      BRG    End
Add:   ADD   B, A
      ADDI  A, 1
      JUMP  Loop
End:   STORE [sum], B
```

**Code Memory:**

```
0011010000000000
0011000000000001
1000110000000000
1101001100000000
1111001000000011
0100010000000000
0101000000000001
1110000011111011
1010010000000010
```

Assembly Language

Machine Language

# Mapping Assembly to Machine Code

**.data**

```
N      BYTE  5
i      BYTE  ?
sum    BYTE  ?
```

**Data Memory:**

```
00000101
00000000
00000000
```

**.code**

```
      LOADI  B, 0
      LOADI  A, 1
      LOAD   D, [N]
Loop:  CMP    A, D
      BRG    End
Add:   ADD   B, A
      ADDI  A, 1
      JUMP  Loop
End:   STORE [sum], B
```

**Code Memory:**

```
00110100_00000000
00110000_00000001
10001100_00000000
11010011_00000000
11110010_00000011
01000100_00000000
01010000_00000001
11100000_11111011
10100100_00000010
```

Assembly Language

Machine Language

# Mapping Assembly to Machine Code

**.data**

```
N      BYTE  5
i      BYTE  ?
sum    BYTE  ?
```

**Data Memory:**

```
00000101
00000000
00000000
```

**.code**

```
      LOADI  B, 0
      LOADI  A, 1
      LOAD   D, [N]
Loop:  CMP    A, D
      BRG    End
Add:   ADD   B, A
      ADDI  A, 1
      JUMP  Loop
End:   STORE [sum], B
```

**Code Memory:**

```
0011_01_00_00000000
0011_00_00_00000001
1000_11_00_00000000
1101_00_11_00000000
1111_00_10_00000011
0100_01_00_00000000
0101_00_00_00000001
1110_00_00_11111011
1010_01_00_00000010
```

Assembly Language

Machine Language

# Mapping Assembly to Machine Code

**.data**

```
N      BYTE  5
i      BYTE  ?
sum    BYTE  ?
```

**Data Memory:**

```
00000101
00000000
00000000
```

**.code**

```
      LOADI  B, 0
      LOADI  A, 1
      LOAD   D, [N]
Loop:  CMP    A, D
      BRG    End
Add:   ADD   B, A
      ADDI  A, 1
      JUMP  Loop
End:   STORE [sum], B
```

**Code Memory:**

```
0011_01_00_00000000
0011_00_00_00000001
1000_11_00_00000000
1101_00_11_00000000
1111_00_10_00000011
0100_01_00_00000000
0101_00_00_00000001
1110_00_00_11111011
1010_01_00_00000010
```

# Mapping Assembly to Machine Code

**.data**

**N**        **BYTE**    **5**  
**i**        **BYTE**    **?**  
**sum**     **BYTE**    **?**

**Data Memory:**

**00000101**  
**00000000**  
**00000000**

**.code**

**LOADI**  **B, 0**  
          **LOADI**  **A, 1**  
          **LOAD**   **D, [N]**  
**Loop:**    **CMP**   **A, D**  
          **BRG**   **End**  
**Add:**    **ADD**   **B, A**  
          **ADDI**  **A, 1**  
          **JUMP**  **Loop**  
**End:**    **STORE**  **[sum], B**

**Code Memory:**

**0011\_01\_00\_00000000**  
**0011\_00\_00\_00000001**  
**1000\_11\_00\_00000000**  
**1101\_00\_11\_00000000**  
**1111\_00\_10\_00000011**  
**0100\_01\_00\_00000000**  
**0101\_00\_00\_00000001**  
**1110\_00\_00\_11111011**  
**1010\_01\_00\_00000010**

# Mapping Assembly to Machine Code

**.data**

```
N      BYTE  5
i      BYTE  ?
sum    BYTE  ?
```

**Data Memory:**

```
00000101
00000000
00000000
```

**.code**

```
      LOADI  B, 0
      LOADI  A, 1
      LOAD   D, [N]
Loop:  CMP    A, D
      BRG    End
Add:   ADD   B, A
      ADDI  A, 1
      JUMP  Loop
End:   STORE [sum], B
```

**Code Memory:**

```
0011_01_00_00000000
0011_00_00_00000001
1000_11_00_00000000
1101_00_11_00000000
1111_00_10_00000011
0100_01_00_00000000
0101_00_00_00000001
1110_00_00_11111011
1010_01_00_00000010
```

# Mapping Assembly to Machine Code

**.data**

**N**        **BYTE**    **5**  
**i**        **BYTE**    **?**  
**sum**     **BYTE**    **?**

**Data Memory:**

00000101  
00000000  
**00000000**

**.code**

**LOADI**   **B, 0**  
      **LOADI**   **A, 1**  
      **LOAD**    **D, [N]**  
**Loop:**   **CMP**     **A, D**  
      **BRG**     **End**  
**Add:**    **ADD**     **B, A**  
      **ADDI**    **A, 1**  
      **JUMP**    **Loop**  
**End:**     **STORE**   **[sum], B**

**Code Memory:**

0011\_01\_00\_00000000  
0011\_00\_00\_00000001  
1000\_11\_00\_00000000  
1101\_00\_11\_00000000  
1111\_00\_10\_00000011  
0100\_01\_00\_00000000  
0101\_00\_00\_00000001  
1110\_00\_00\_11111011  
1010\_01\_00\_00000010



# OPCODE Mapping

**.data**

**N**        **BYTE**    **5**  
**i**        **BYTE**    **?**  
**sum**     **BYTE**    **?**

**Data Memory:**

00000101  
00000000  
00000000

**.code**

**LOADI**    **B, 0**  
          **LOADI**    **A, 1**  
          **LOAD**     **D, [N]**  
**Loop:**    **CMP**      **A, D**  
          **BRG**      **End**  
**Add:**    **ADD**      **B, A**  
          **ADDI**     **A, 1**  
          **JUMP**     **Loop**  
**End:**     **STORE**    **[sum], B**

**Code Memory:**

0011\_01\_00\_00000000  
0011\_00\_00\_00000001  
1000\_11\_00\_00000000  
1101\_00\_11\_00000000  
1111\_00\_10\_00000011  
0100\_01\_00\_00000000  
0101\_00\_00\_00000001  
1110\_00\_00\_11111011  
1010\_01\_00\_00000010

# OPCODE Mapping

**.data**

**N**        **BYTE**    **5**  
**i**        **BYTE**    **?**  
**sum**     **BYTE**    **?**

**Data Memory:**

**00000101**  
**00000000**  
**00000000**

**.code**

**LOADI**    **B, 0**  
          **LOADI**    **A, 1**  
          **LOAD**     **D, [N]**  
**Loop:**    **CMP**        **A, D**  
          **BRG**        **End**  
**Add:**    **ADD**        **B, A**  
          **ADDI**     **A, 1**  
          **JUMP**     **Loop**  
**End:**     **STORE**    **[sum], B**

**Code Memory:**

**0011\_01\_00\_00000000**  
**0011\_00\_00\_00000001**  
**1000\_11\_00\_00000000**  
**1101\_00\_11\_00000000**  
**1111\_00\_10\_00000011**  
**0100\_01\_00\_00000000**  
**0101\_00\_00\_00000001**  
**1110\_00\_00\_11111011**  
**1010\_01\_00\_00000010**

# Register Parameter Mapping

**.data**

<b>N</b>	<b>BYTE</b>	<b>5</b>
<b>i</b>	<b>BYTE</b>	<b>?</b>
<b>sum</b>	<b>BYTE</b>	<b>?</b>

**Data Memory:**

<b>00000101</b>
<b>00000000</b>
<b>00000000</b>

**.code**

	<b>LOADI</b>	<b>B, 0</b>
	<b>LOADI</b>	<b>A, 1</b>
	<b>LOAD</b>	<b>D, [N]</b>
<b>Loop:</b>	<b>CMP</b>	<b>A, D</b>
	<b>BRG</b>	<b>End</b>
<b>Add:</b>	<b>ADD</b>	<b>B, A</b>
	<b>ADDI</b>	<b>A, 1</b>
	<b>JUMP</b>	<b>Loop</b>
<b>End:</b>	<b>STORE</b>	<b>[sum], B</b>

**Code Memory:**

<b>0011_01_00_00000000</b>
<b>0011_00_00_00000001</b>
<b>1000_11_00_00000000</b>
<b>1101_00_11_00000000</b>
<b>1111_00_10_00000011</b>
<b>0100_01_00_00000000</b>
<b>0101_00_00_00000001</b>
<b>1110_00_00_11111011</b>
<b>1010_01_00_00000010</b>

# Register Parameter Mapping

**.data**

<b>N</b>	<b>BYTE</b>	<b>5</b>
<b>i</b>	<b>BYTE</b>	<b>?</b>
<b>sum</b>	<b>BYTE</b>	<b>?</b>

**Data Memory:**

<b>00000101</b>
<b>00000000</b>
<b>00000000</b>

**.code**

	<b>LOADI</b>	<b>B, 0</b>
	<b>LOADI</b>	<b>A, 1</b>
	<b>LOAD</b>	<b>D, [N]</b>
<b>Loop:</b>	<b>CMP</b>	<b>A, D</b>
	<b>BRG</b>	<b>End</b>
<b>Add:</b>	<b>ADD</b>	<b>B, A</b>
	<b>ADDI</b>	<b>A, 1</b>
	<b>JUMP</b>	<b>Loop</b>
<b>End:</b>	<b>STORE</b>	<b>[sum], B</b>

**Code Memory:**

<b>0011_01_00_00000000</b>
<b>0011_00_00_00000001</b>
<b>1000_11_00_00000000</b>
<b>1101_00_11_00000000</b>
<b>1111_00_10_00000011</b>
<b>0100_01_00_00000000</b>
<b>0101_00_00_00000001</b>
<b>1110_00_00_11111011</b>
<b>1010_01_00_00000010</b>

# Second Register Parameter Mapping

**.data**

<b>N</b>	<b>BYTE</b>	<b>5</b>
<b>i</b>	<b>BYTE</b>	<b>?</b>
<b>sum</b>	<b>BYTE</b>	<b>?</b>

**Data Memory:**

<b>00000101</b>
<b>00000000</b>
<b>00000000</b>

**.code**

	<b>LOADI</b>	<b>B, 0</b>
	<b>LOADI</b>	<b>A, 1</b>
	<b>LOAD</b>	<b>D, [N]</b>
<b>Loop:</b>	<b>CMP</b>	<b>A, D</b>
	<b>BRG</b>	<b>End</b>
<b>Add:</b>	<b>ADD</b>	<b>B, A</b>
	<b>ADDI</b>	<b>A, 1</b>
	<b>JUMP</b>	<b>Loop</b>
<b>End:</b>	<b>STORE</b>	<b>[sum], B</b>

**Code Memory:**

<b>0011_01_00_00000000</b>
<b>0011_00_00_00000001</b>
<b>1000_11_00_00000000</b>
<b>1101_00_11_00000000</b>
<b>1111_00_10_00000011</b>
<b>0100_01_00_00000000</b>
<b>0101_00_00_00000001</b>
<b>1110_00_00_11111011</b>
<b>1010_01_00_00000010</b>

# Second Register Parameter Mapping

**.data**

**N**        **BYTE**    **5**  
**i**        **BYTE**    **?**  
**sum**     **BYTE**    **?**

**Data Memory:**

00000101  
00000000  
00000000

**.code**

**LOADI** **B**, **0**  
          **LOADI** **A**, **1**  
          **LOAD**  **D**, [**N**]  
**Loop:**    **CMP**  **A**, **D**  
          **BRG**  **End**  
**Add:**    **ADD**  **B**, **A**  
          **ADDI** **A**, **1**  
          **JUMP** **Loop**  
**End:**     **STORE** [**sum**], **B**

**Code Memory:**

0011\_01\_00\_00000000  
0011\_00\_00\_00000001  
1000\_11\_00\_00000000  
1101\_00\_11\_00000000  
1111\_00\_10\_00000011  
0100\_01\_00\_00000000  
0101\_00\_00\_00000001  
1110\_00\_00\_11111011  
1010\_01\_00\_00000010

# Value / Address / Offset Mapping

**.data**

<b>N</b>	<b>BYTE</b>	<b>5</b>
<b>i</b>	<b>BYTE</b>	<b>?</b>
<b>sum</b>	<b>BYTE</b>	<b>?</b>

**Data Memory:**

<b>00000101</b>
<b>00000000</b>
<b>00000000</b>

**.code**

	<b>LOADI</b>	<b>B, 0</b>
	<b>LOADI</b>	<b>A, 1</b>
	<b>LOAD</b>	<b>D, [N]</b>
<b>Loop:</b>	<b>CMP</b>	<b>A, D</b>
	<b>BRG</b>	<b>End</b>
<b>Add:</b>	<b>ADD</b>	<b>B, A</b>
	<b>ADDI</b>	<b>A, 1</b>
	<b>JUMP</b>	<b>Loop</b>
<b>End:</b>	<b>STORE</b>	<b>[sum], B</b>

**Code Memory:**

<b>0011_01_00_00000000</b>
<b>0011_00_00_00000001</b>
<b>1000_11_00_00000000</b>
<b>1101_00_11_00000000</b>
<b>1111_00_10_00000011</b>
<b>0100_01_00_00000000</b>
<b>0101_00_00_00000001</b>
<b>1110_00_00_11111011</b>
<b>1010_01_00_00000010</b>

# Value / Address / Offset Mapping

**.data**

<b>N</b>	<b>BYTE</b>	<b>5</b>
<b>i</b>	<b>BYTE</b>	<b>?</b>
<b>sum</b>	<b>BYTE</b>	<b>?</b>

**Data Memory:**

<b>00000101</b>
<b>00000000</b>
<b>00000000</b>

**.code**

	<b>LOADI</b>	<b>B, 0</b>
	<b>LOADI</b>	<b>A, 1</b>
	<b>LOAD</b>	<b>D, [N]</b>
<b>Loop:</b>	<b>CMP</b>	<b>A, D</b>
	<b>BRG</b>	<b>End</b>
<b>Add:</b>	<b>ADD</b>	<b>B, A</b>
	<b>ADDI</b>	<b>A, 1</b>
	<b>JUMP</b>	<b>Loop</b>
<b>End:</b>	<b>STORE</b>	<b>[sum], B</b>

**Code Memory:**

<b>0011_01_00_00000000</b>
<b>0011_00_00_00000001</b>
<b>1000_11_00_00000000</b>
<b>1101_00_11_00000000</b>
<b>1111_00_10_00000011</b>
<b>0100_01_00_00000000</b>
<b>0101_00_00_00000001</b>
<b>1110_00_00_11111011</b>
<b>1010_01_00_00000010</b>



# “Don’t care” bits ...

**.data**

```
N      BYTE  5
i      BYTE  ?
sum    BYTE  ?
```

**Data Memory:**

```
00000101
00000000
00000000
```

**.code**

```
      LOADI  B, 0
      LOADI  A, 1
      LOAD   D, [N]
Loop:  CMP    A, D
      BRG    End
Add:   ADD   B, A
      ADDI  A, 1
      JUMP  Loop
End:   STORE [sum], B
```

**Code Memory:**

```
0011_01_dd_00000000
0011_00_dd_00000001
1000_11_dd_00000000
1101_00_11_dddddddd
1111_dd_10_00000011
0100_01_00_dddddddd
0101_00_dd_00000001
1110_dd_dd_11110111
1010_01_dd_00000010
```

# ... are mapped to 0 by the Assembler

**.data**

```
N      BYTE  5
i      BYTE  ?
sum    BYTE  ?
```

**Data Memory:**

```
00000101
00000000
00000000
```

**.code**

```
      LOADI B, 0
      LOADI A, 1
      LOAD  D, [N]
Loop: CMP   A, D
      BRG   End
Add: ADD   B, A
      ADDI  A, 1
      JUMP  Loop
End:  STORE [sum], B
```

**Code Memory:**

```
0011_01_00_00000000
0011_00_00_00000001
1000_11_00_00000000
1101_00_11_00000000
1111_00_10_00000011
0100_01_00_00000000
0101_00_00_00000001
1110_00_00_11111011
1010_01_00_00000010
```

# Mapping Assembly to Machine Code

**.data**

**N**        **BYTE**    **5**  
**i**        **BYTE**    **?**  
**sum**     **BYTE**    **?**

**Data Memory:**

**00000101**  
**00000000**  
**00000000**

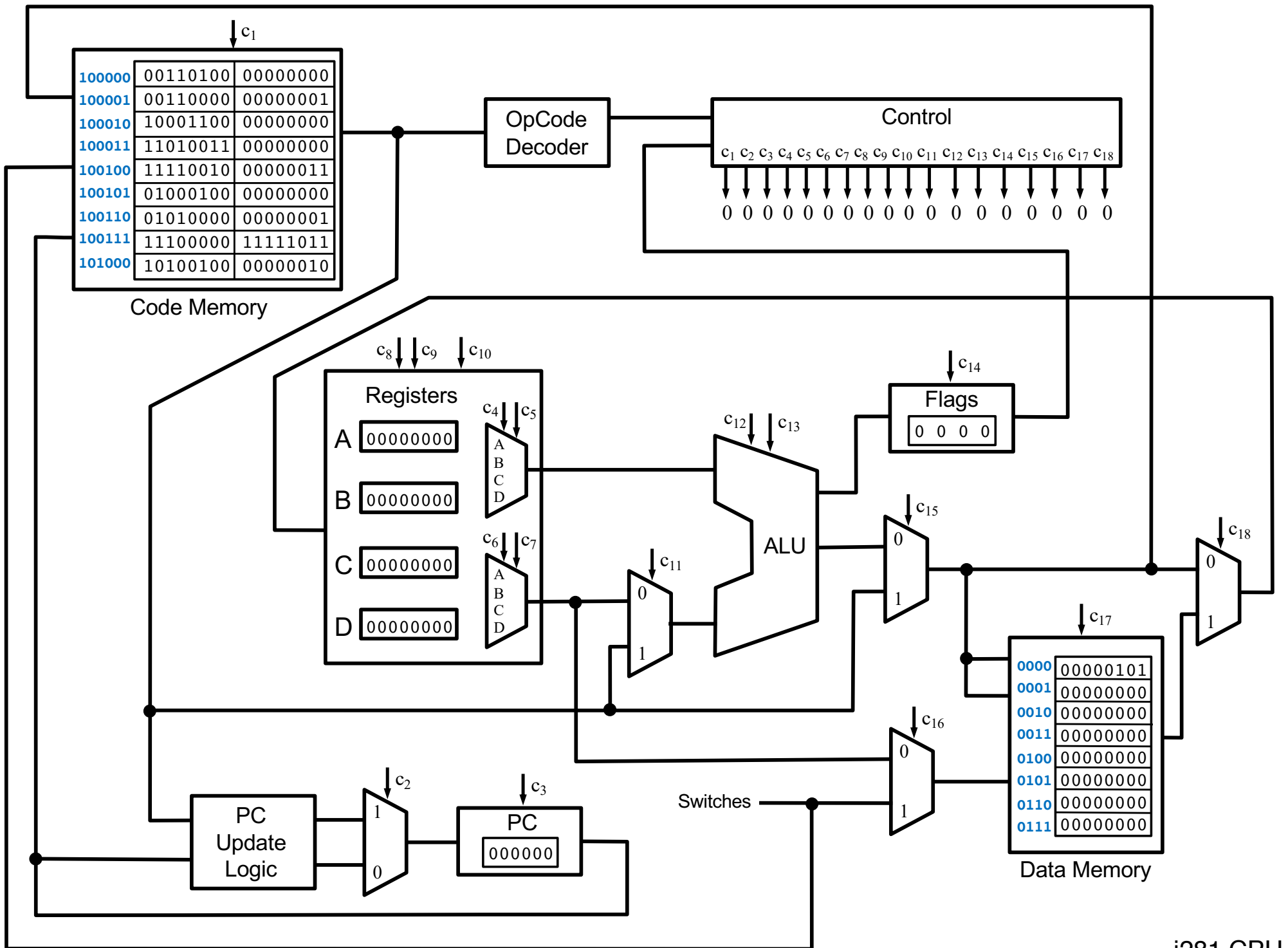
**.code**

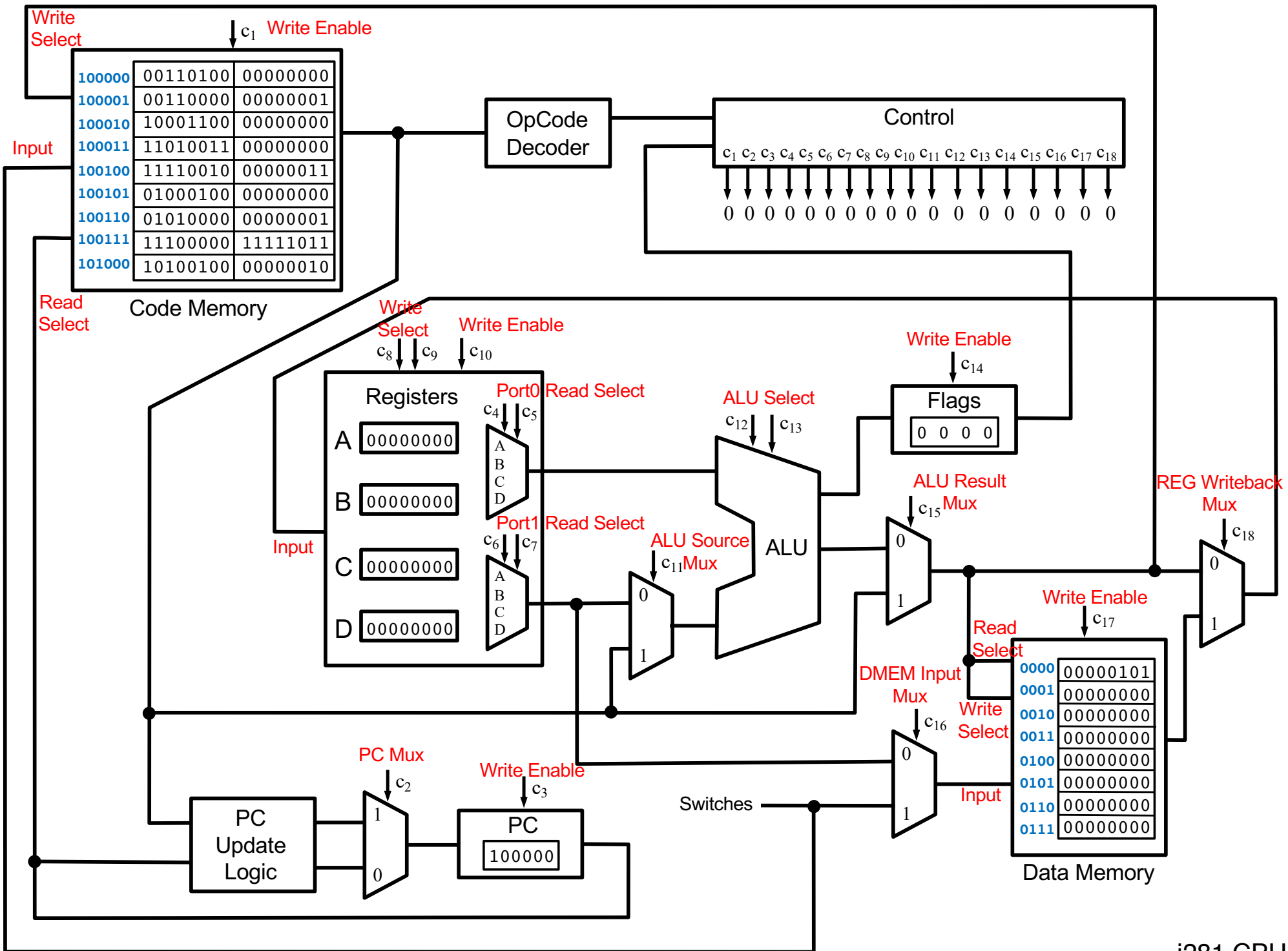
**LOADI** **B, 0**  
          **LOADI** **A, 1**  
          **LOAD** **D, [N]**  
**Loop:**    **CMP** **A, D**  
          **BRG** **End**  
**Add:**    **ADD** **B, A**  
          **ADDI** **A, 1**  
          **JUMP** **Loop**  
**End:**     **STORE** **[sum], B**

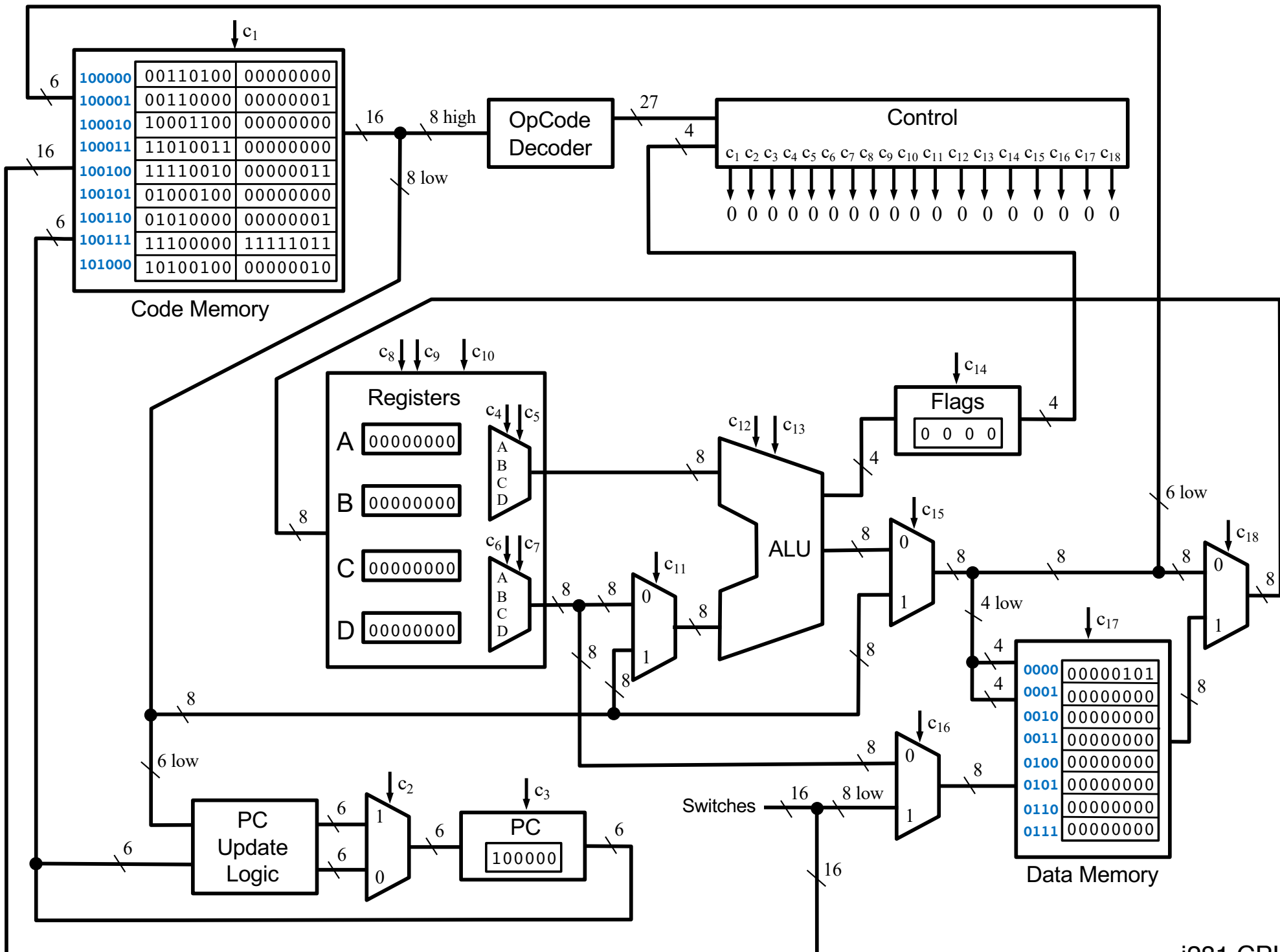
**Code Memory:**

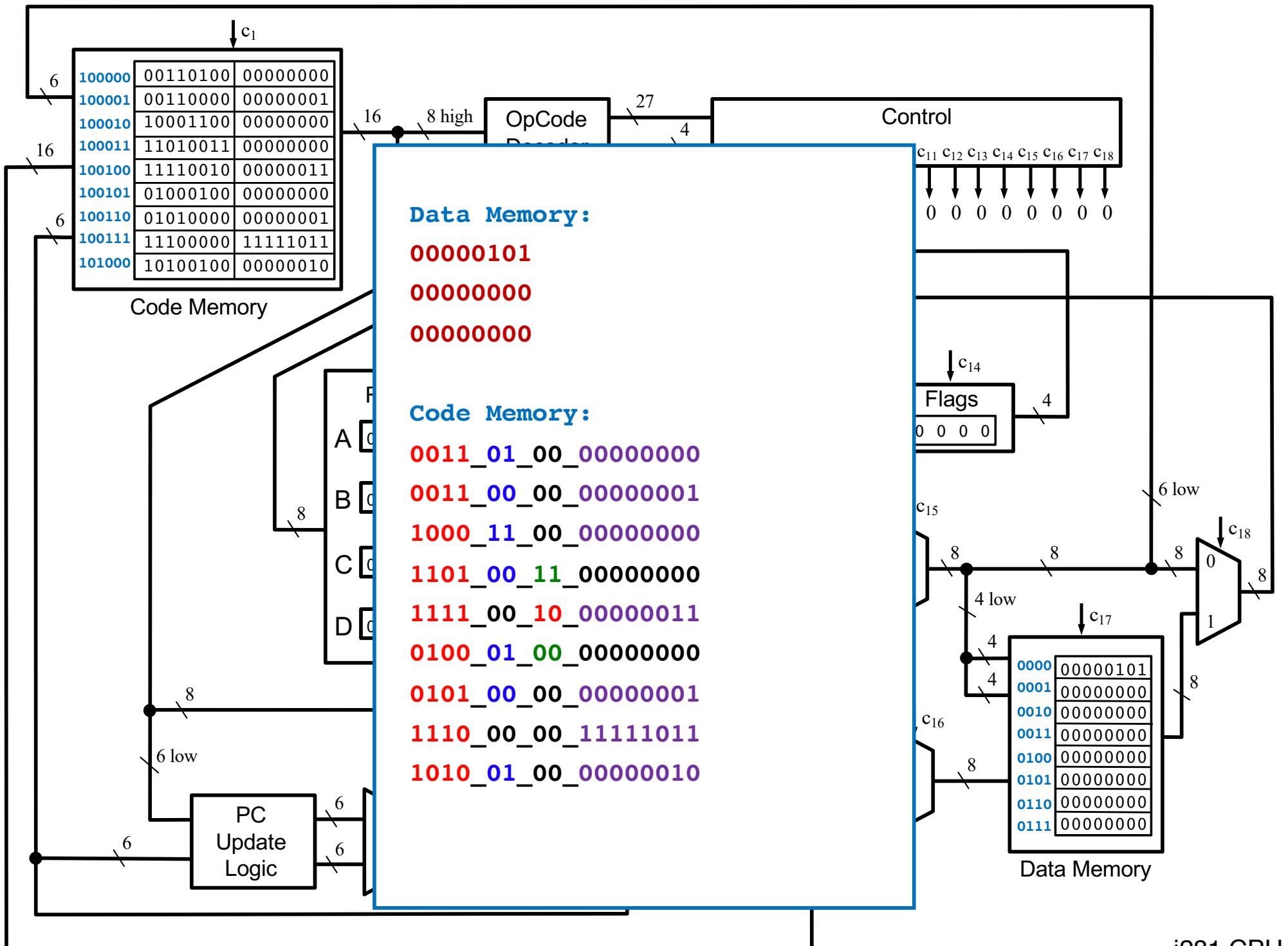
**0011\_01\_00\_00000000**  
**0011\_00\_00\_00000001**  
**1000\_11\_00\_00000000**  
**1101\_00\_11\_00000000**  
**1111\_00\_10\_00000011**  
**0100\_01\_00\_00000000**  
**0101\_00\_00\_00000001**  
**1110\_00\_00\_11111011**  
**1010\_01\_00\_00000010**

# **Loading the Program into Memory**

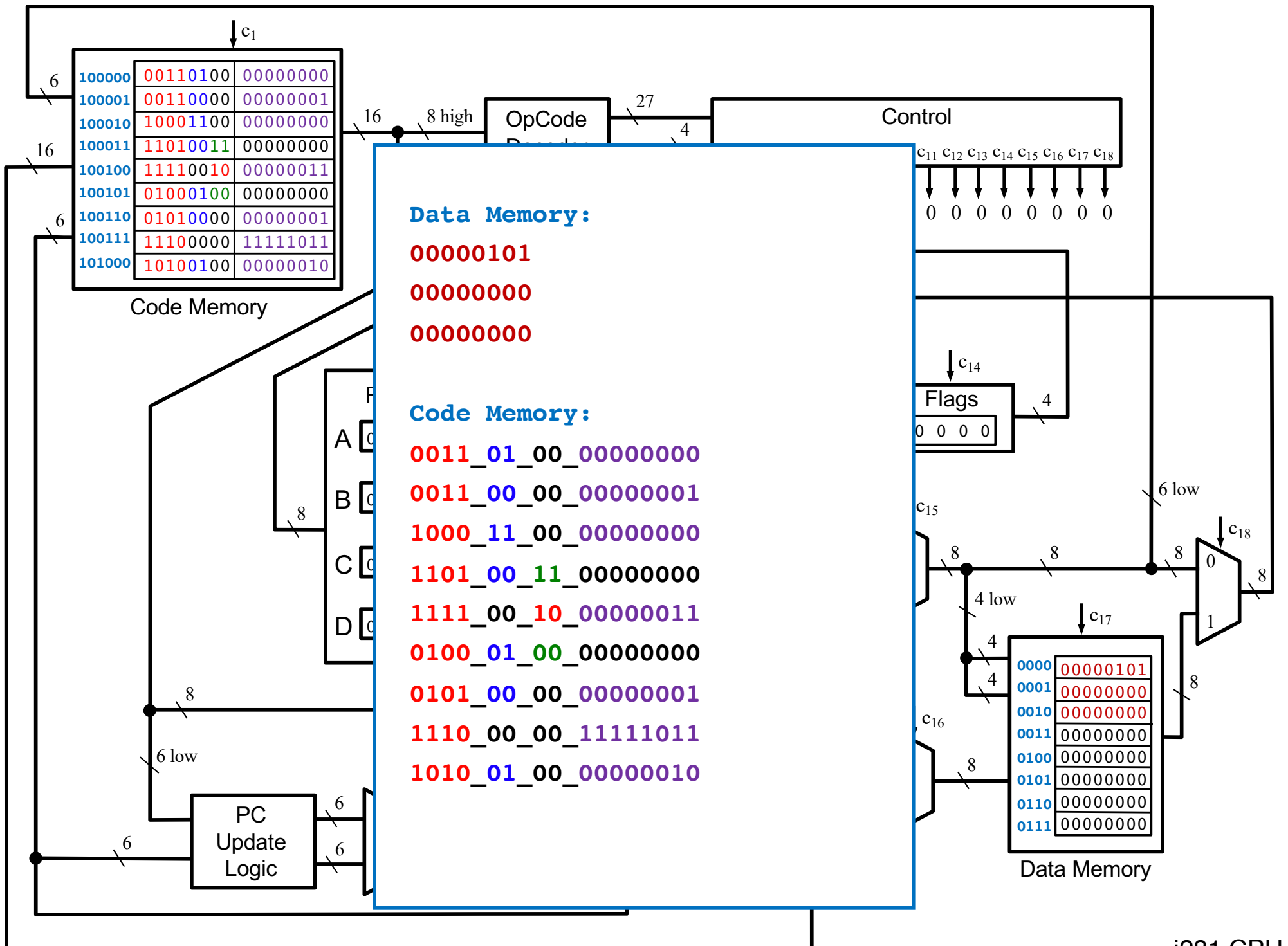


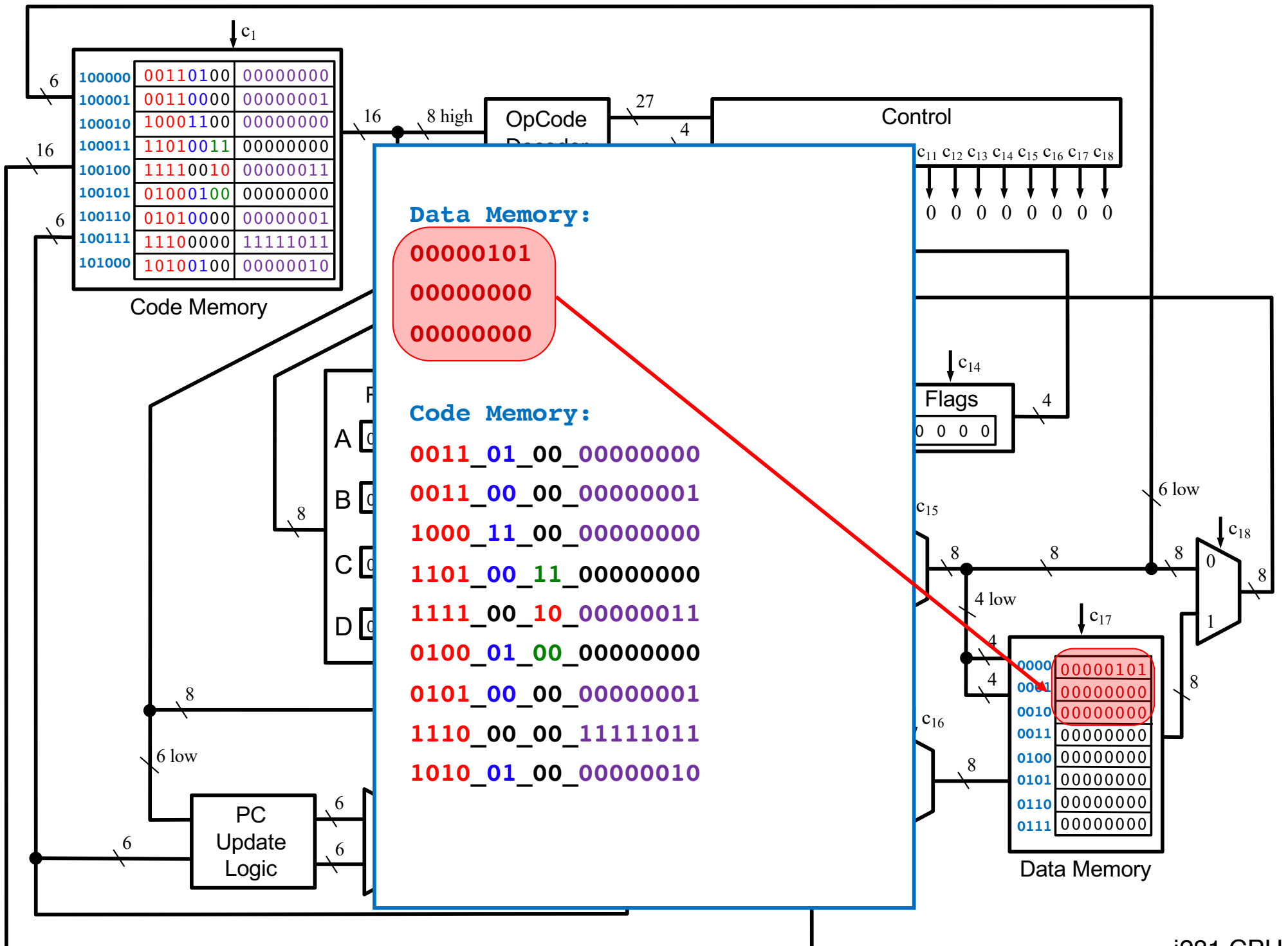


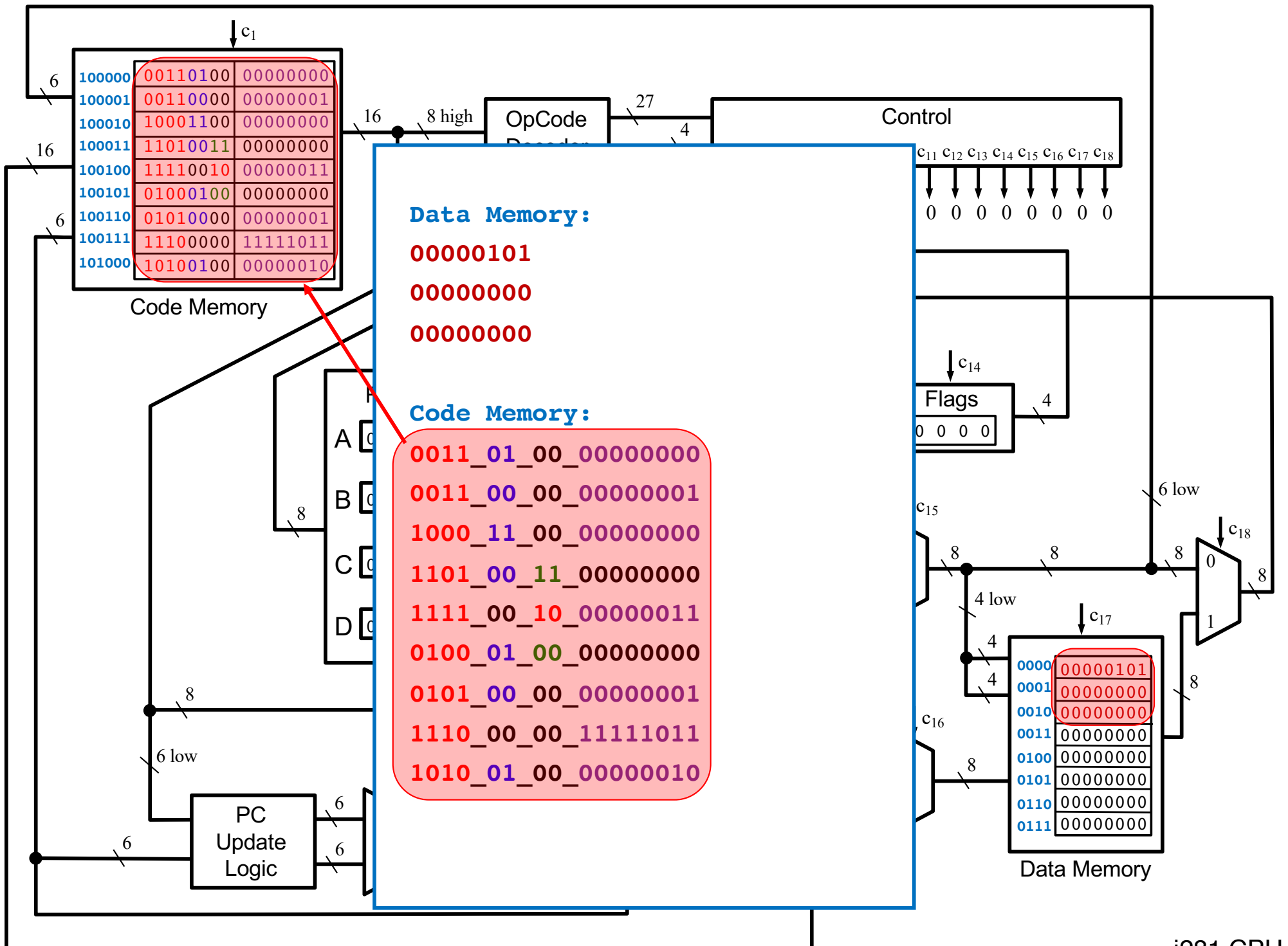


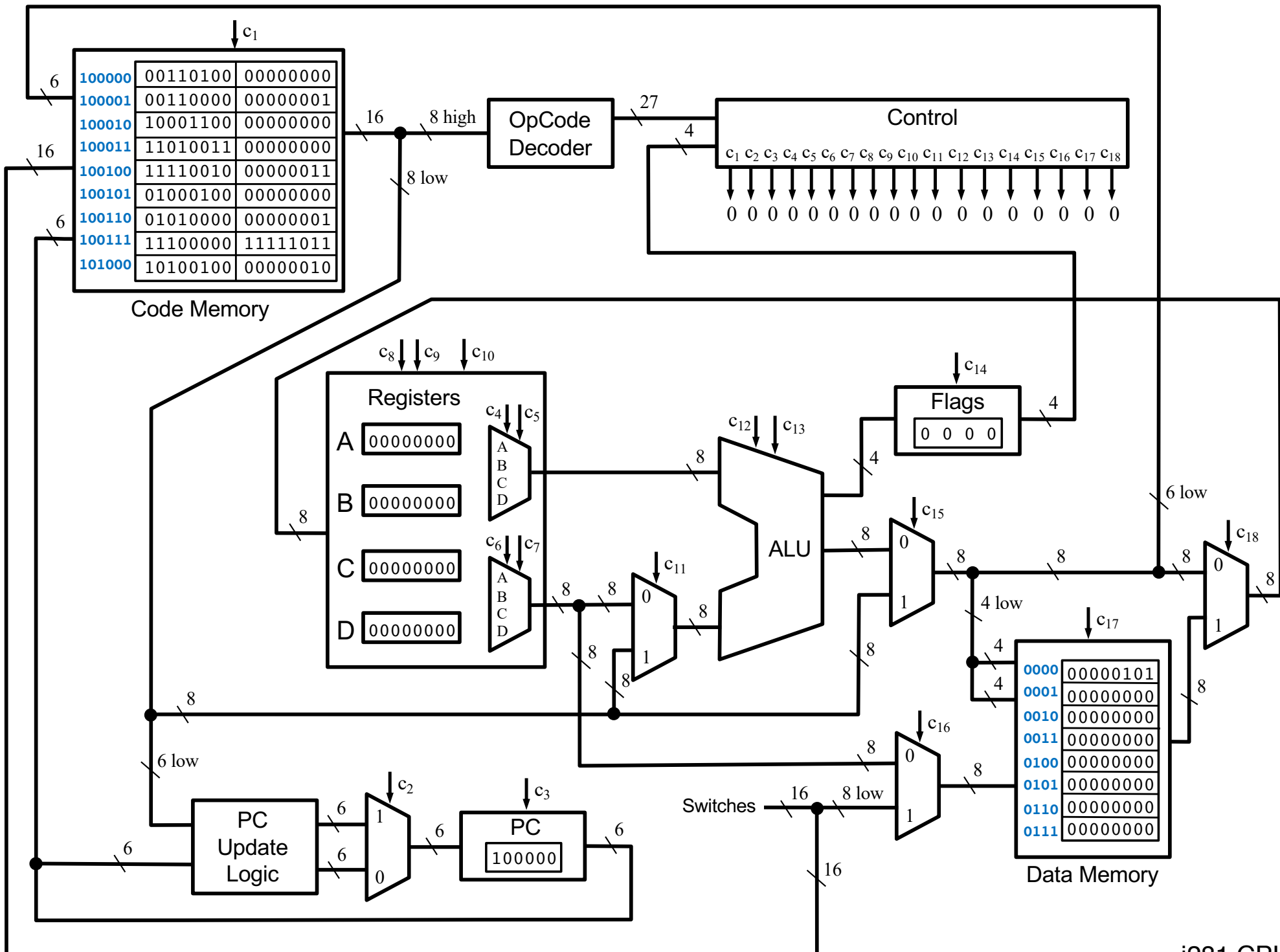




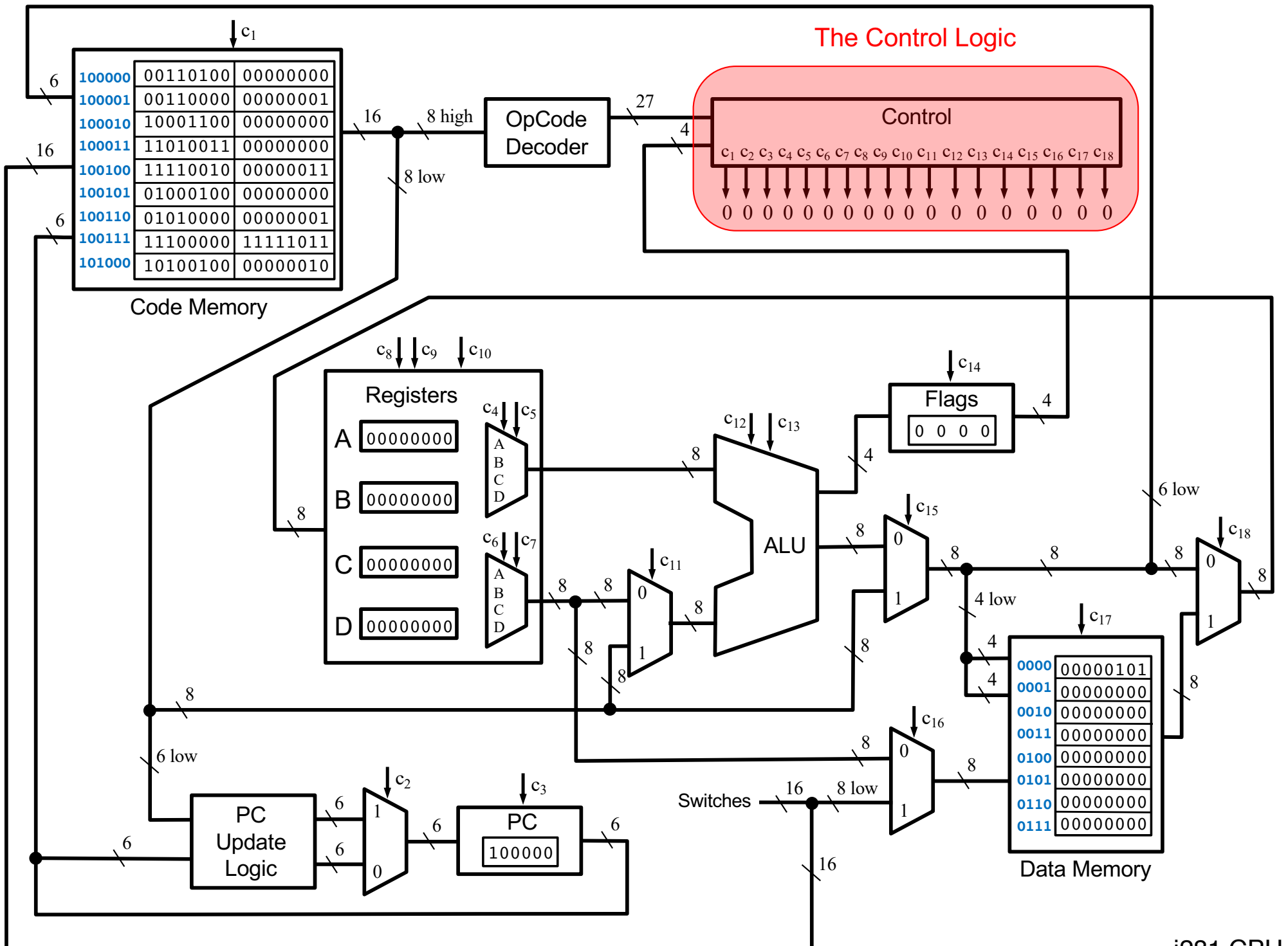


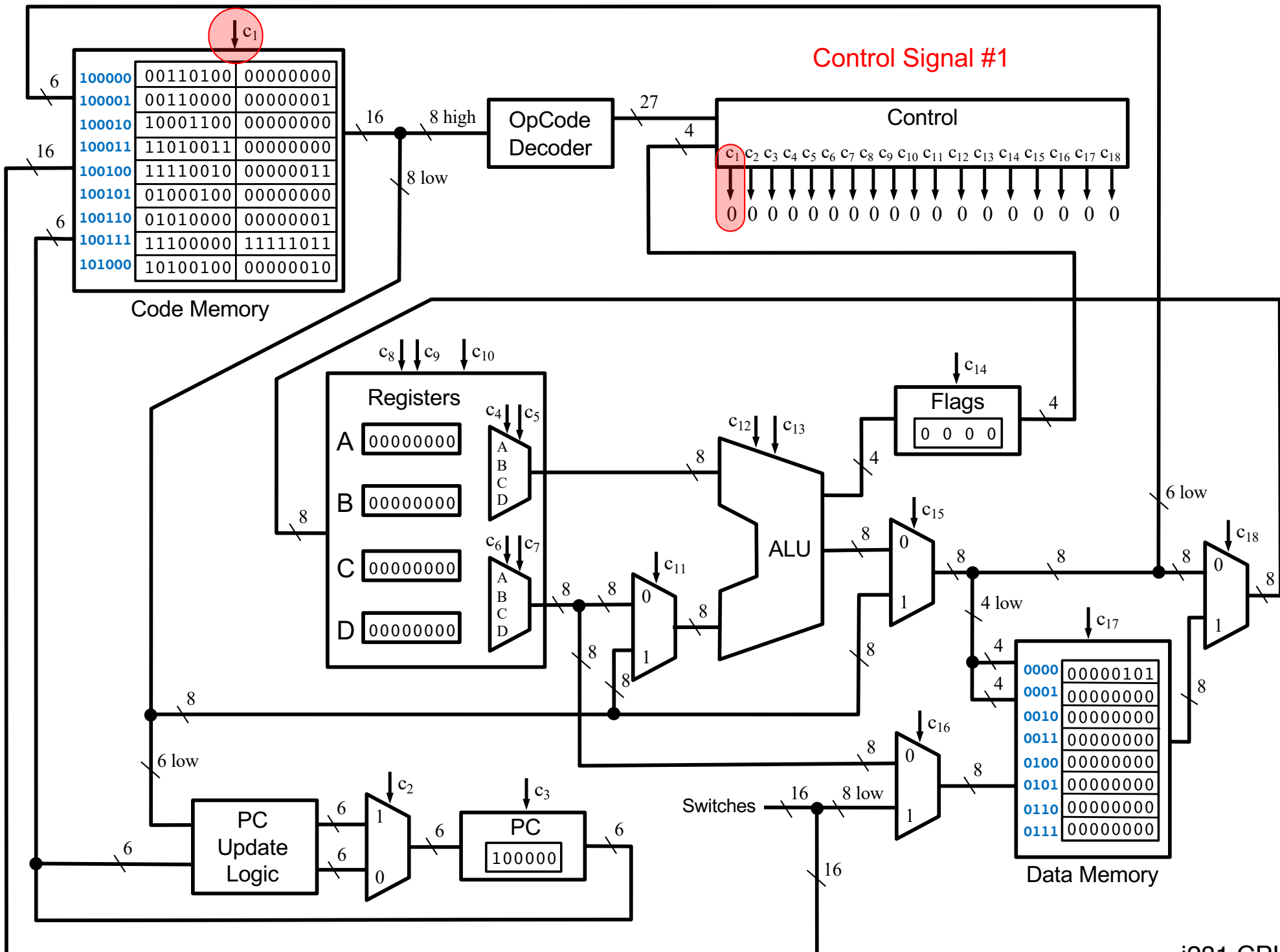


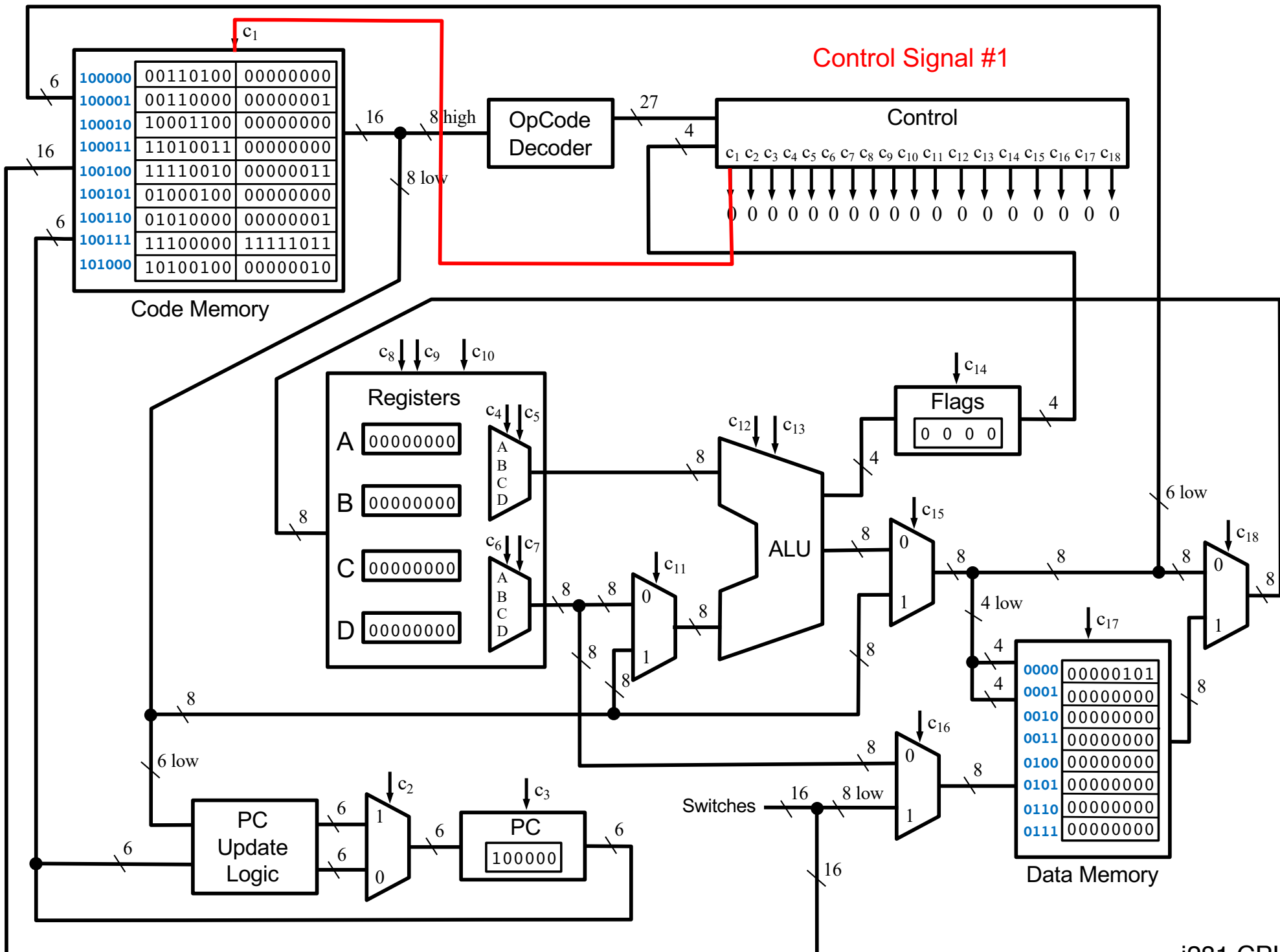




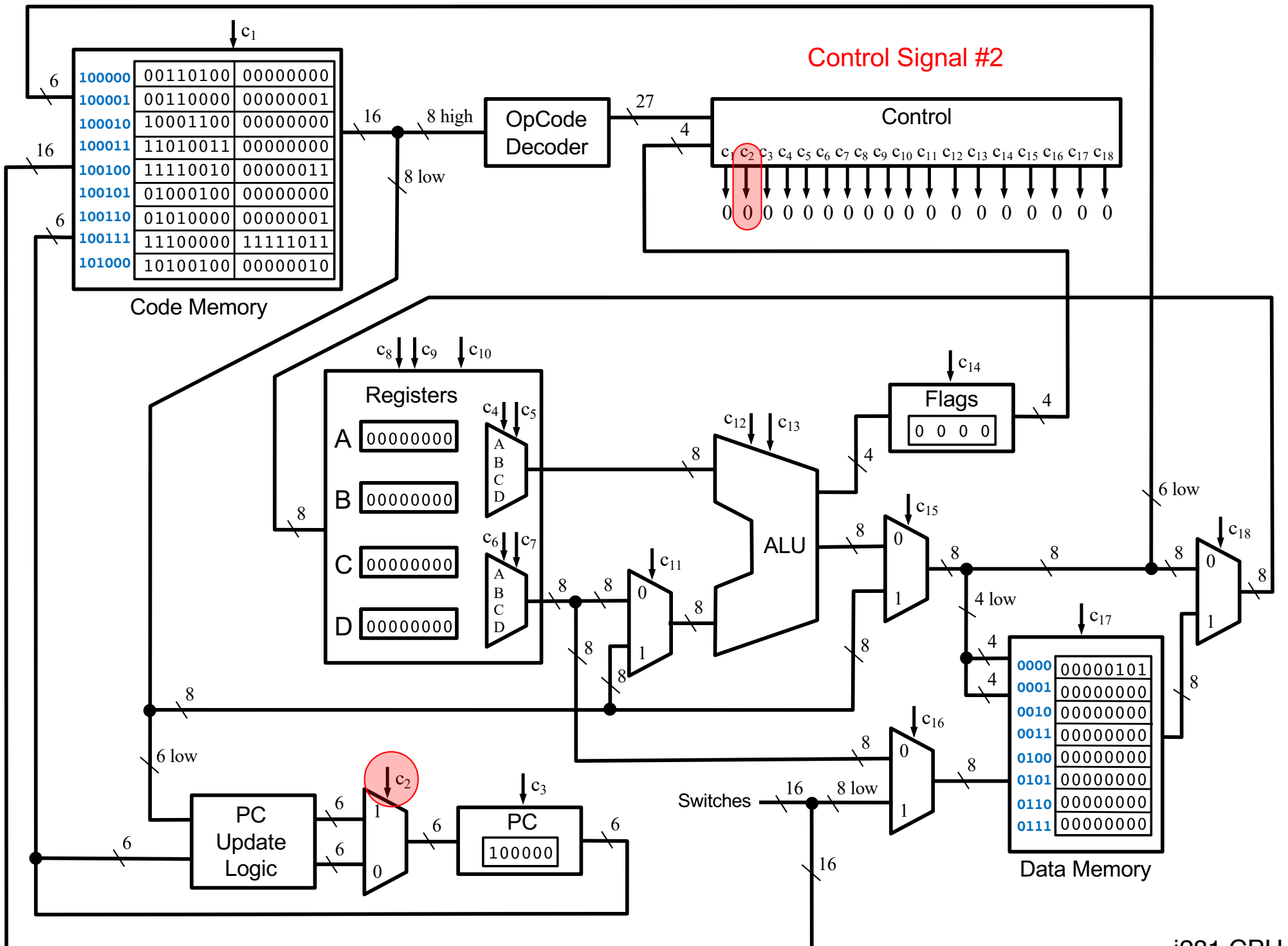
# **The CPU Control Logic**

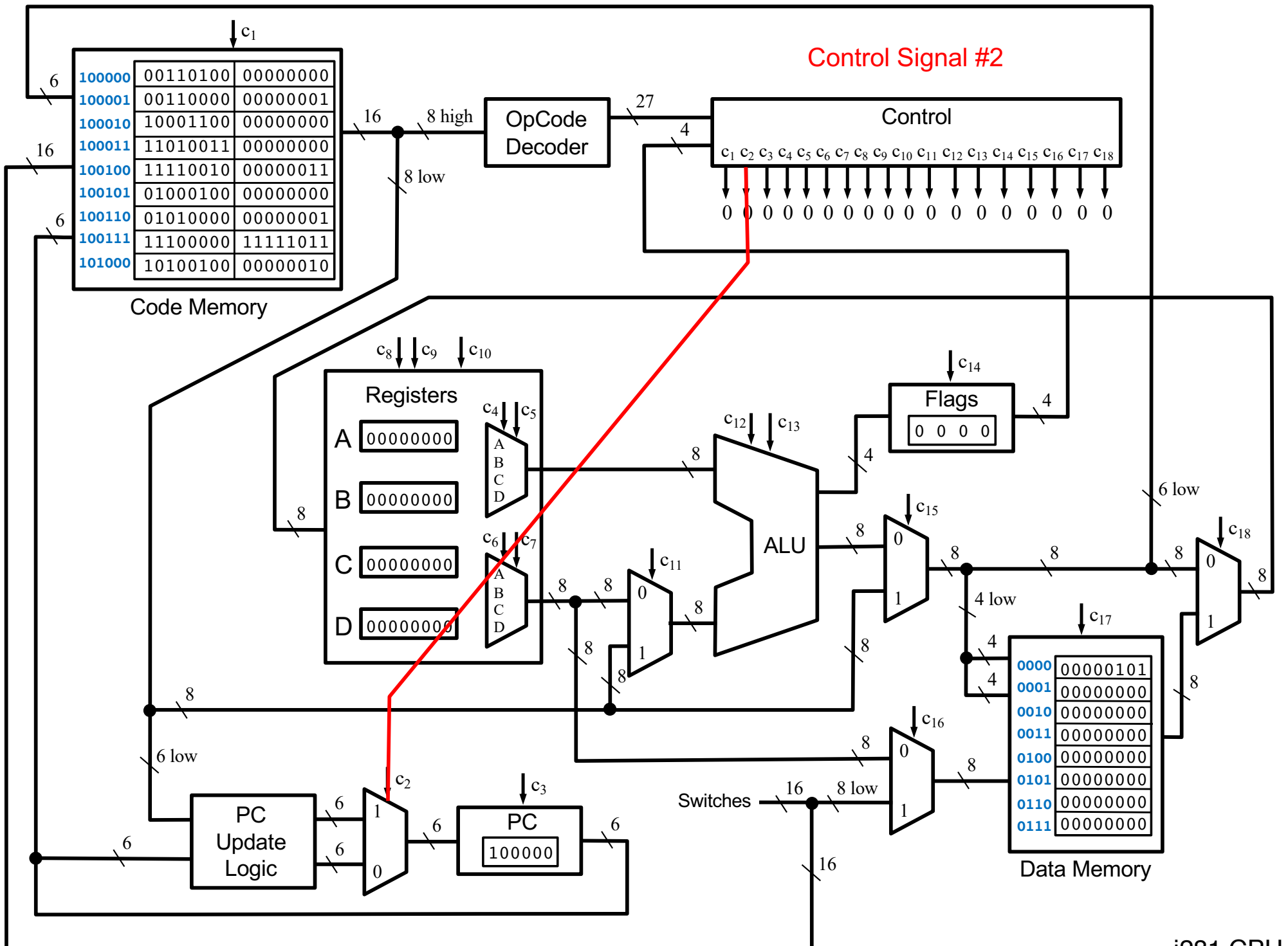


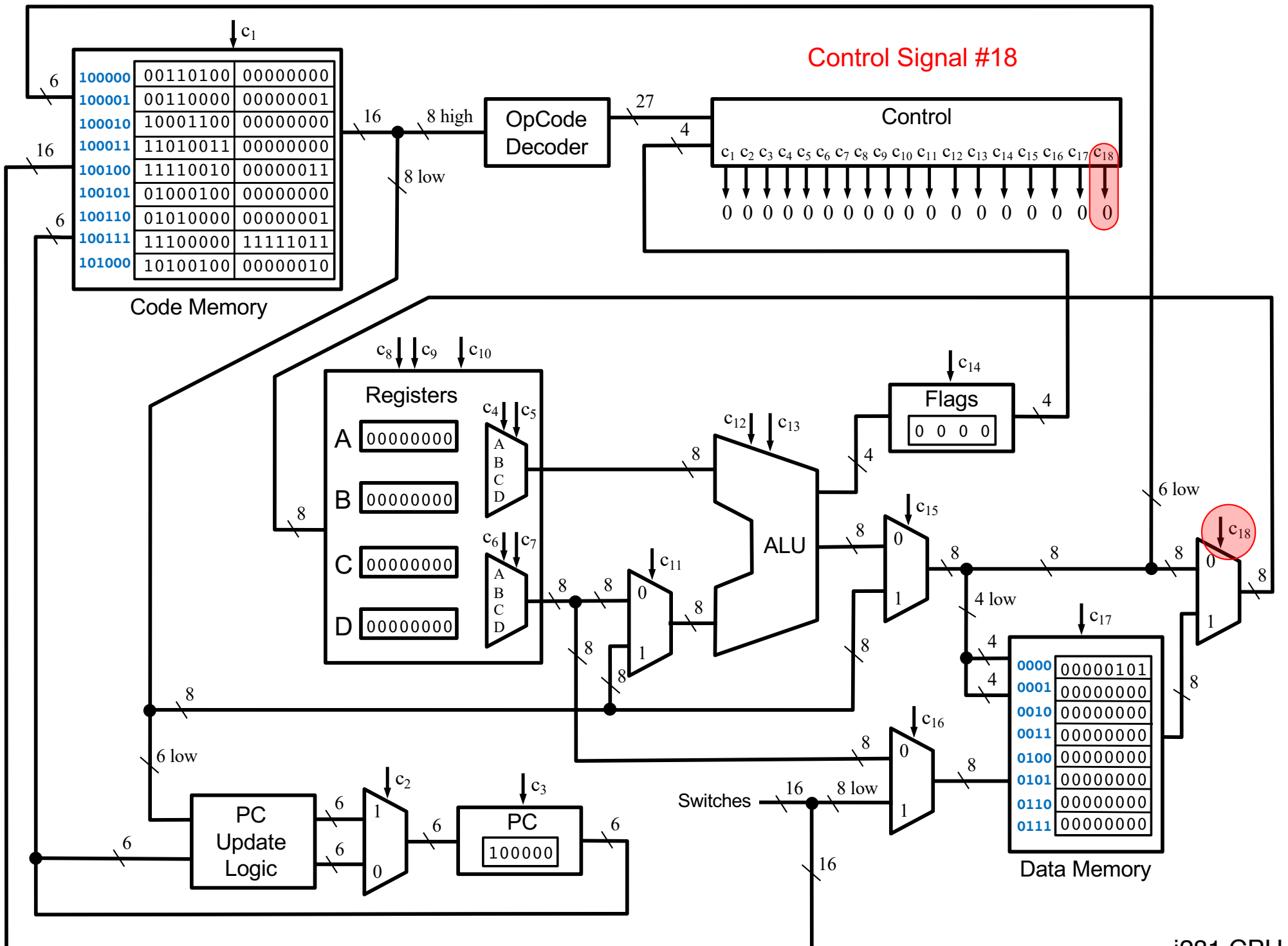


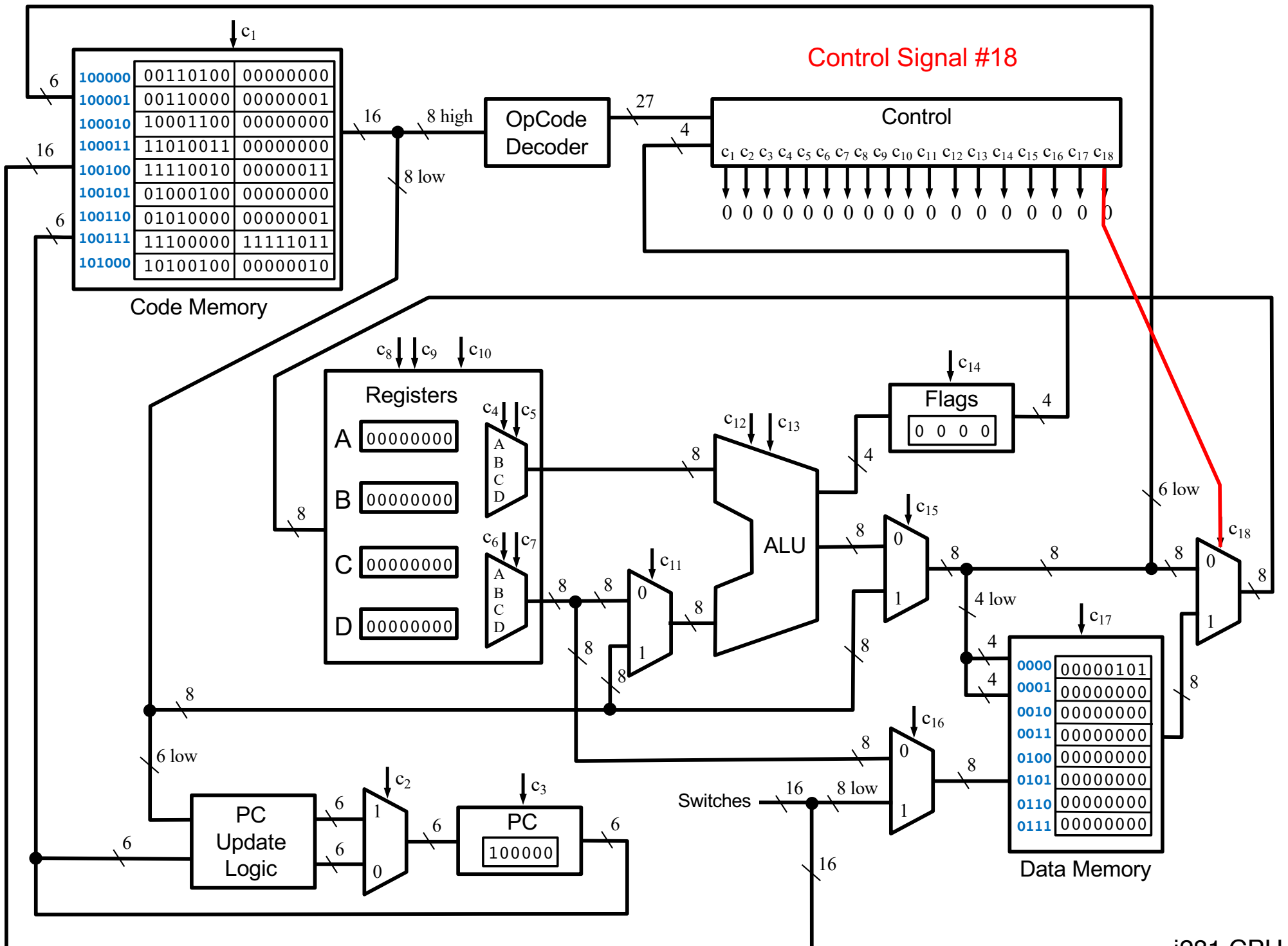


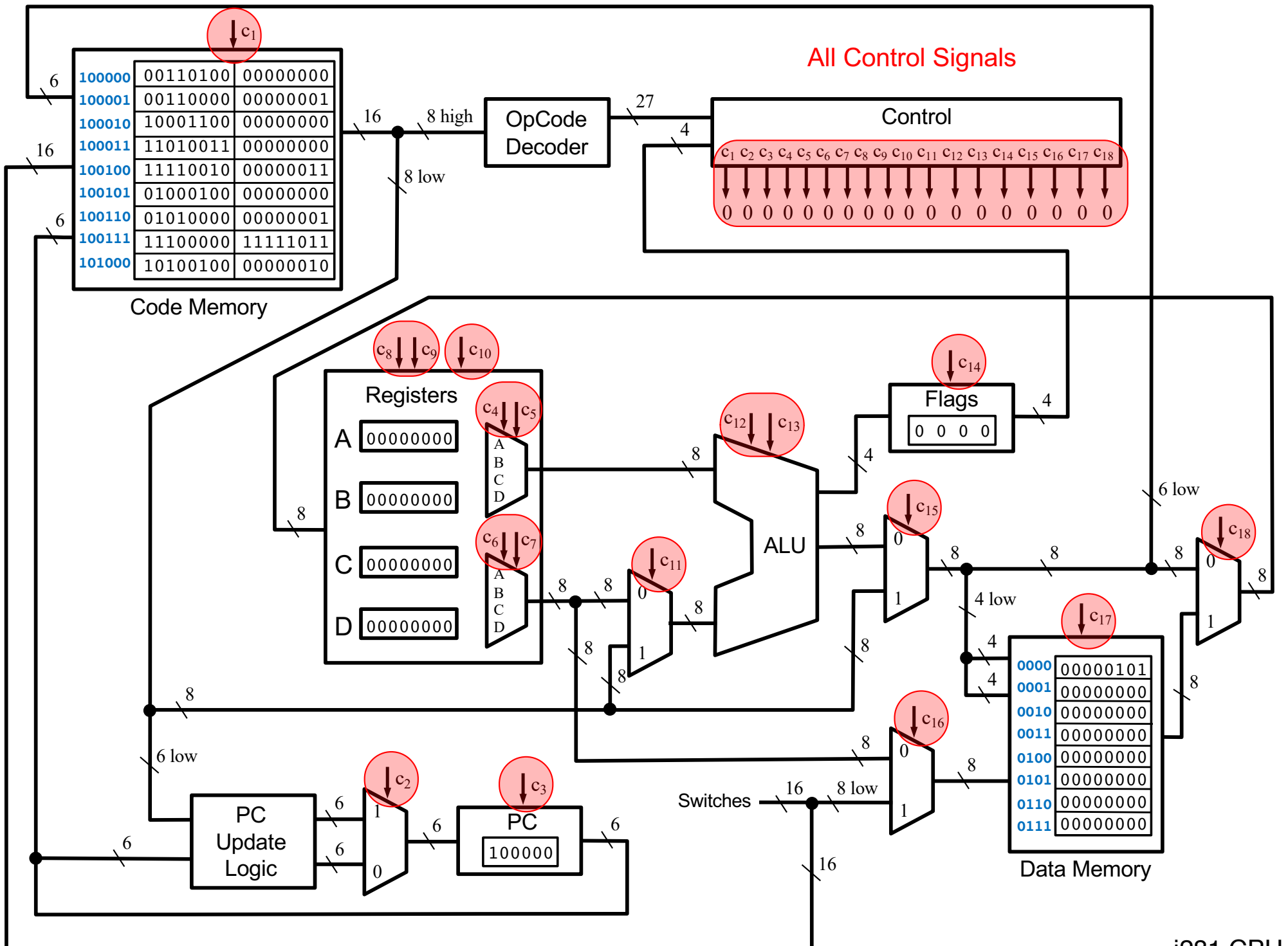


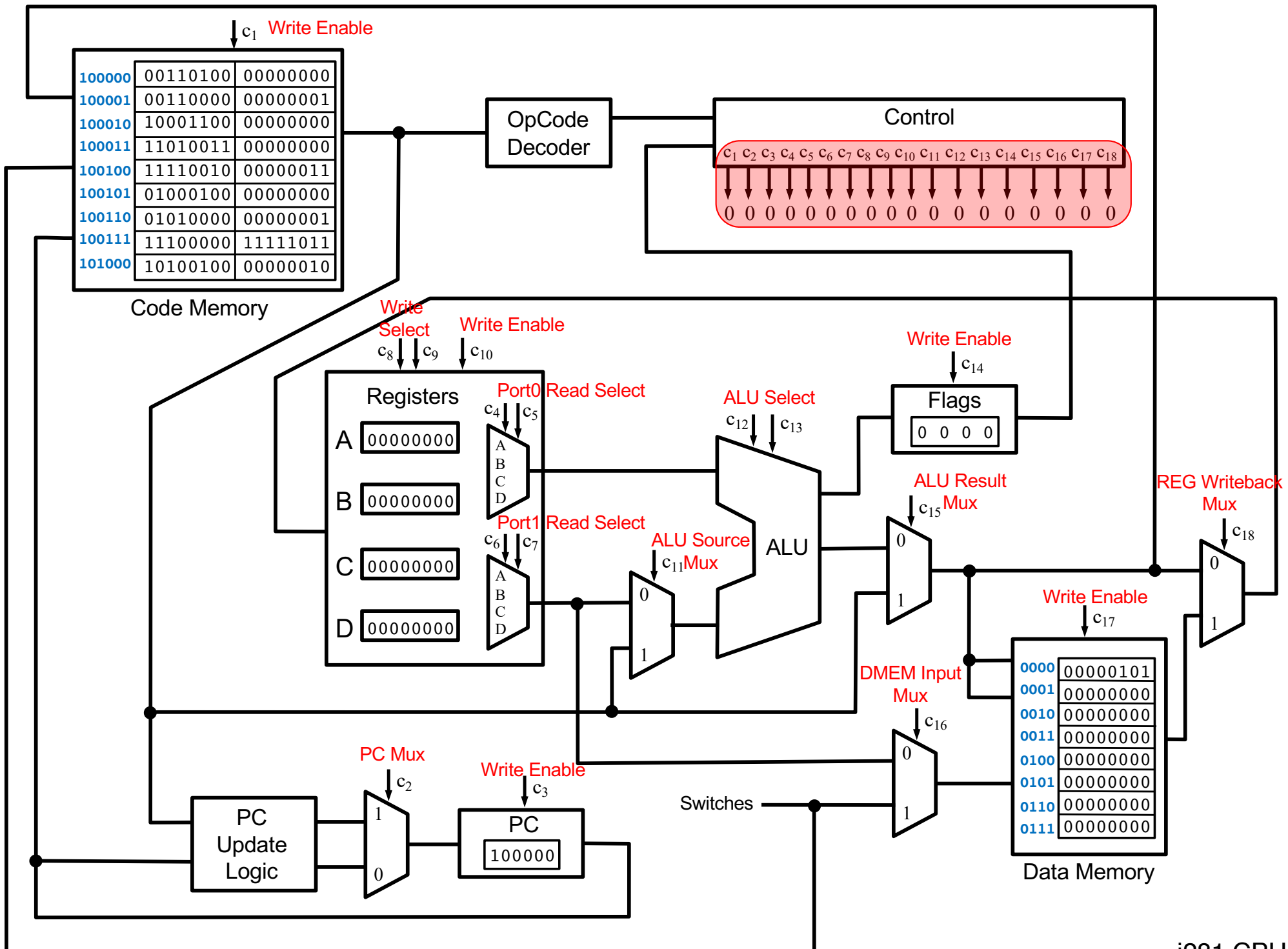


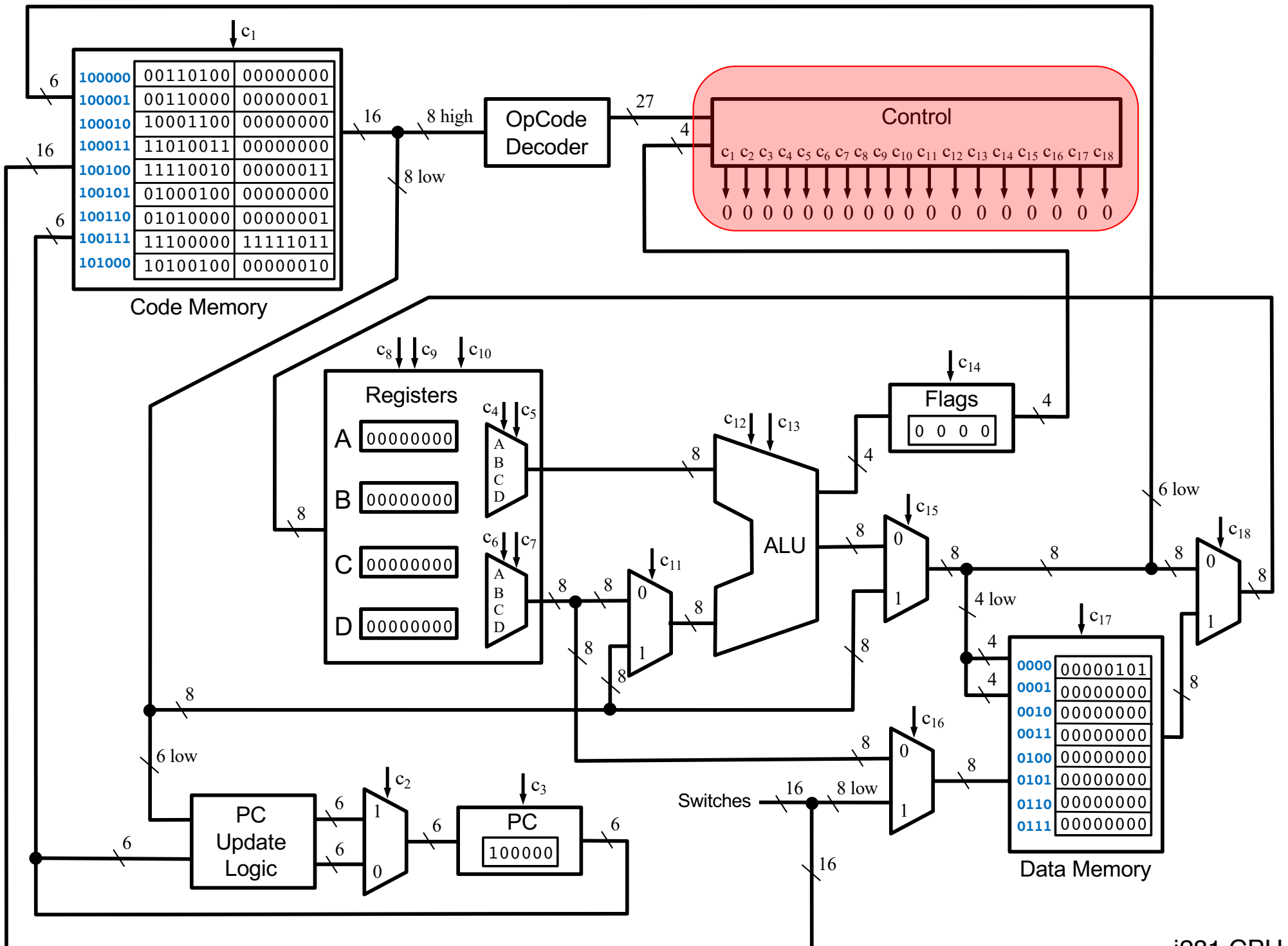












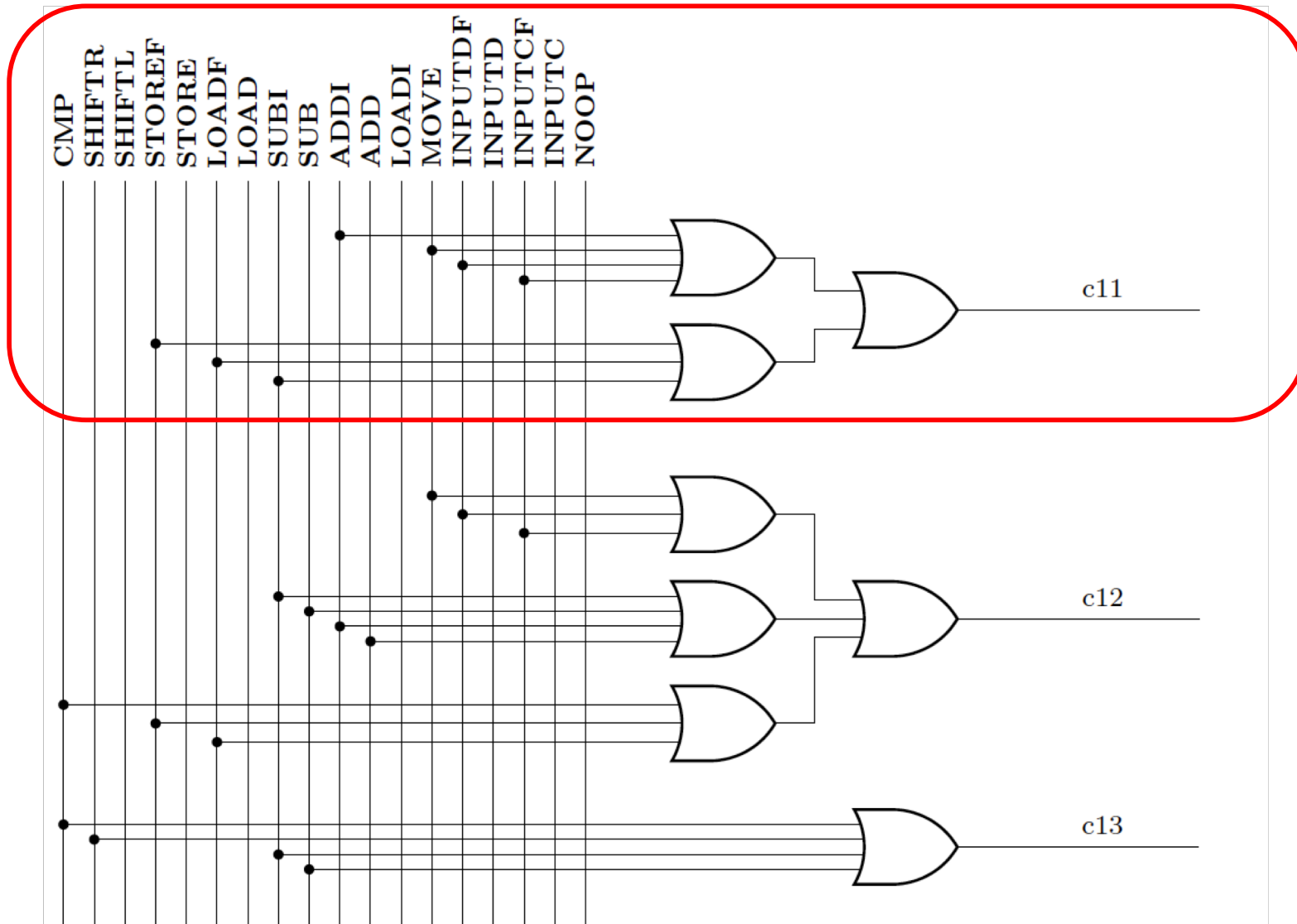






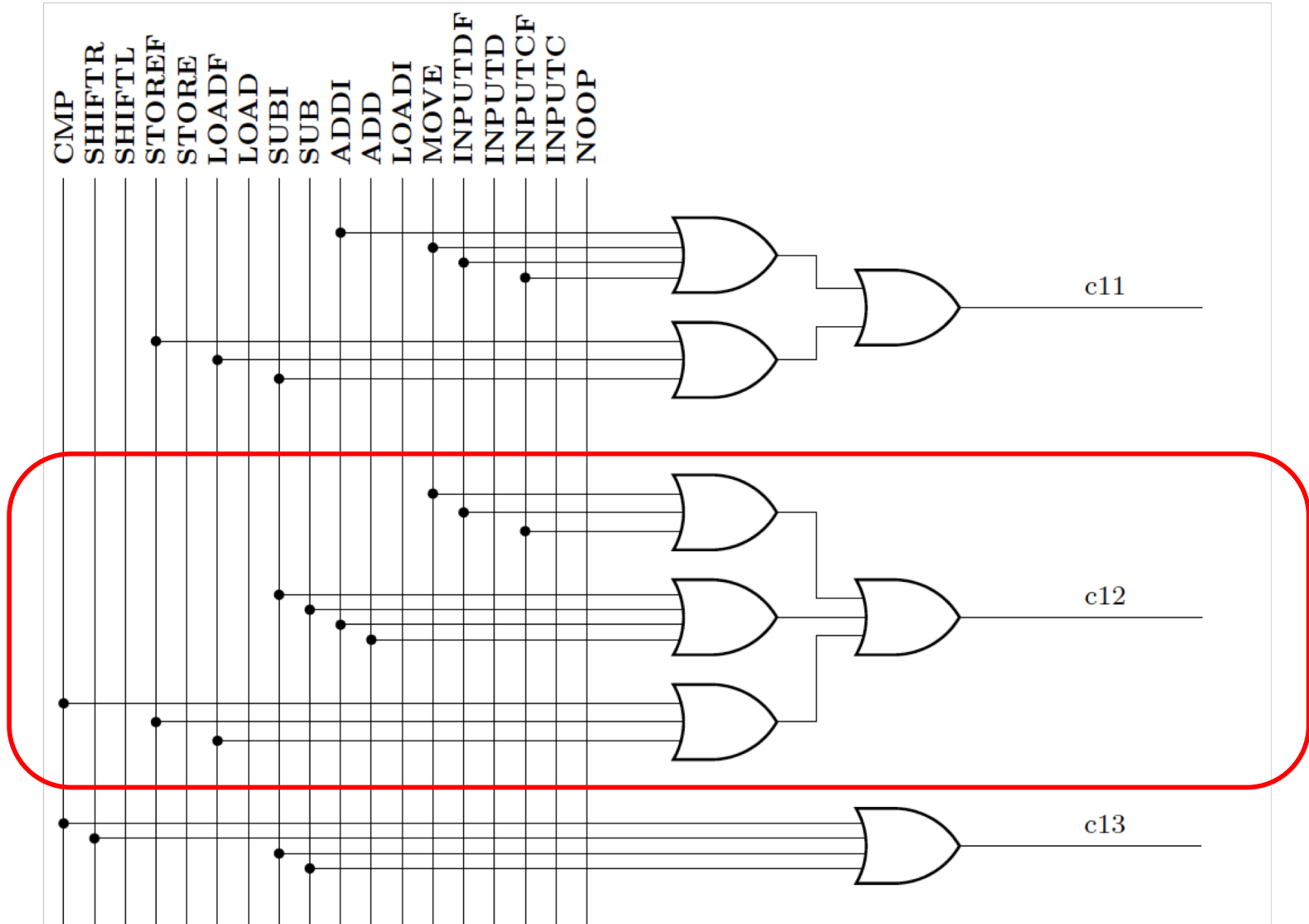


# The Wiring Diagram for $c_{11}$



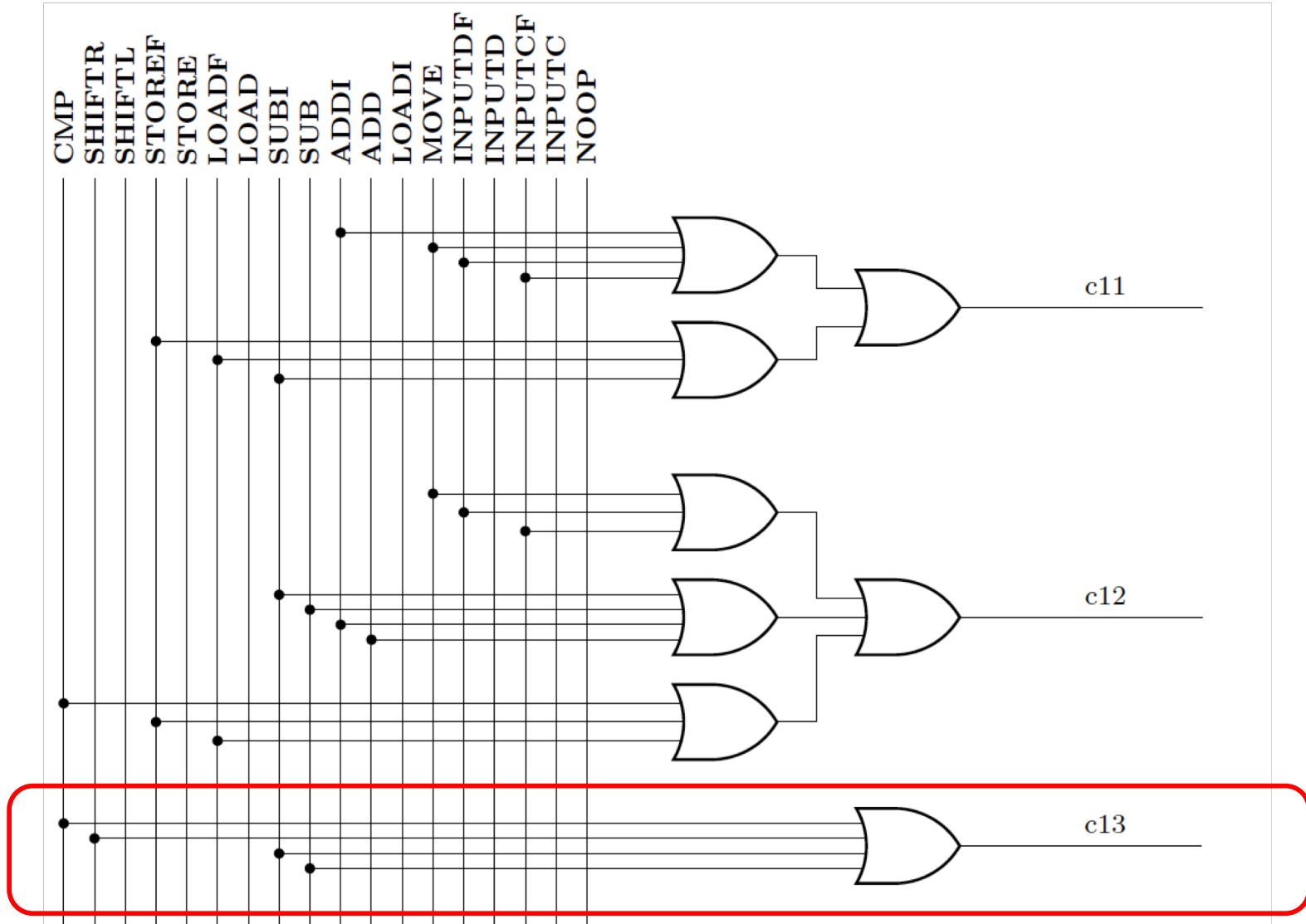


# The Wiring Diagram for $c_{12}$





# The Wiring Diagram for $c_{13}$













# **Simulation of the Program Execution**

# Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

# Mapping Assembly to Machine Code

**.data**

**N**        **BYTE**    **5**  
**i**        **BYTE**    **?**  
**sum**     **BYTE**    **?**

**Data Memory:**

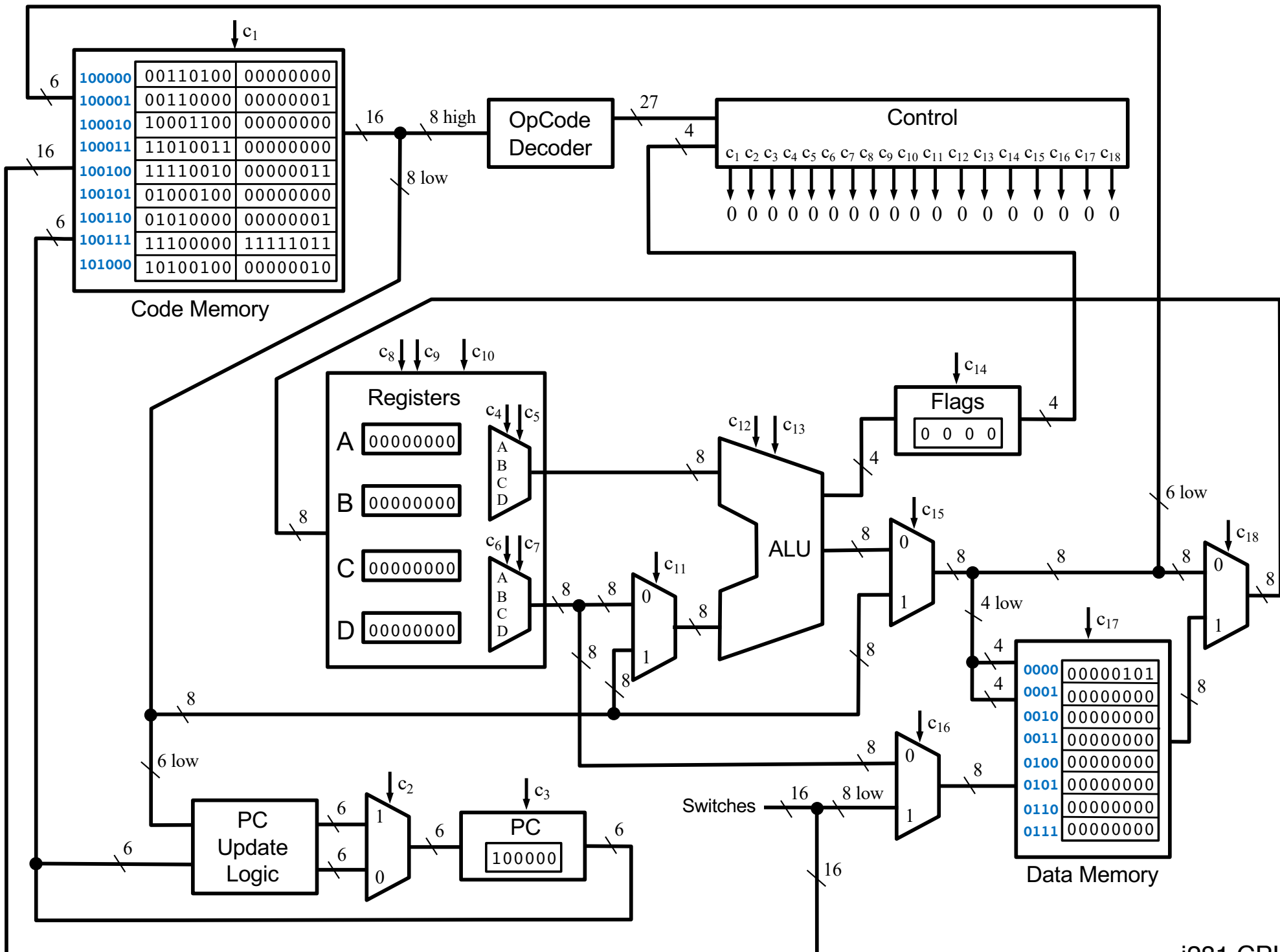
**00000101**  
**00000000**  
**00000000**

**.code**

**LOADI** **B, 0**  
          **LOADI** **A, 1**  
          **LOAD** **D, [N]**  
**Loop:**    **CMP** **A, D**  
          **BRG** **End**  
**Add:**    **ADD** **B, A**  
          **ADDI** **A, 1**  
          **JUMP** **Loop**  
**End:**     **STORE** **[sum], B**

**Code Memory:**

**0011\_01\_00\_00000000**  
**0011\_00\_00\_00000001**  
**1000\_11\_00\_00000000**  
**1101\_00\_11\_00000000**  
**1111\_00\_10\_00000011**  
**0100\_01\_00\_00000000**  
**0101\_00\_00\_00000001**  
**1110\_00\_00\_11111011**  
**1010\_01\_00\_00000010**



LOADI B, 0

100000	00110100	00000000
100001	00110000	00000001
100010	10001100	00000000
100011	11010011	00000000
100100	11110010	00000011
100101	01000100	00000000
100110	01010000	00000001
100111	11110000	11111011
101000	10100100	00000010

Code Memory

OpCode Decoder

Control

c <sub>1</sub>	c <sub>2</sub>	c <sub>3</sub>	c <sub>4</sub>	c <sub>5</sub>	c <sub>6</sub>	c <sub>7</sub>	c <sub>8</sub>	c <sub>9</sub>	c <sub>10</sub>	c <sub>11</sub>	c <sub>12</sub>	c <sub>13</sub>	c <sub>14</sub>	c <sub>15</sub>	c <sub>16</sub>	c <sub>17</sub>	c <sub>18</sub>	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Registers

A: 00000000

B: 00000000

C: 00000000

D: 00000000

Selection: A, B, C, D

ALU

Selection: 0, 1

Flags

0 0 0 0

Data Memory

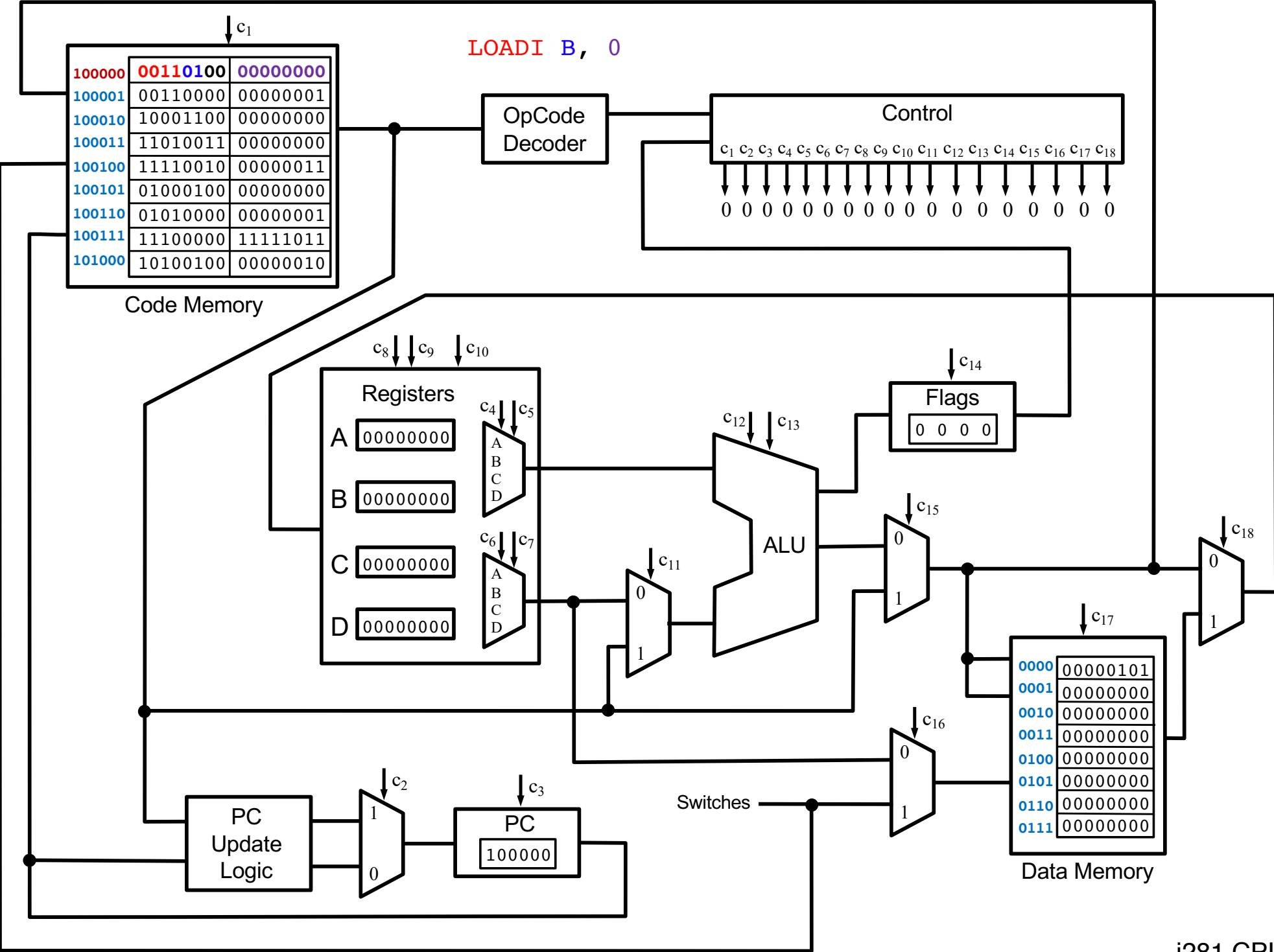
0000	00000101
0001	00000000
0010	00000000
0011	00000000
0100	00000000
0101	00000000
0110	00000000
0111	00000000

PC Update Logic

PC

100000

Switches





LOADI B, 0

100000	00110100	00000000
100001	00110000	00000001
100010	10001100	00000000
100011	11010011	00000000
100100	11110010	00000011
100101	01000100	00000000
100110	01010000	00000001
100111	11100000	11111011
101000	10100100	00000010

Code Memory

OpCode Decoder

Control

c <sub>1</sub>	c <sub>2</sub>	c <sub>3</sub>	c <sub>4</sub>	c <sub>5</sub>	c <sub>6</sub>	c <sub>7</sub>	c <sub>8</sub>	c <sub>9</sub>	c <sub>10</sub>	c <sub>11</sub>	c <sub>12</sub>	c <sub>13</sub>	c <sub>14</sub>	c <sub>15</sub>	c <sub>16</sub>	c <sub>17</sub>	c <sub>18</sub>	
0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0

Registers

A	00000000
B	00000000
C	00000000
D	00000000

c<sub>4</sub> ↓ c<sub>5</sub> ↓  
 A B C D  
 c<sub>6</sub> ↓ c<sub>7</sub> ↓  
 A B C D

ALU

c<sub>12</sub> ↓ c<sub>13</sub> ↓

Flags

0	0	0	0
---	---	---	---

c<sub>14</sub> ↓

Data Memory

0000	00000101
0001	00000000
0010	00000000
0011	00000000
0100	00000000
0101	00000000
0110	00000000
0111	00000000

c<sub>17</sub> ↓

PC Update Logic

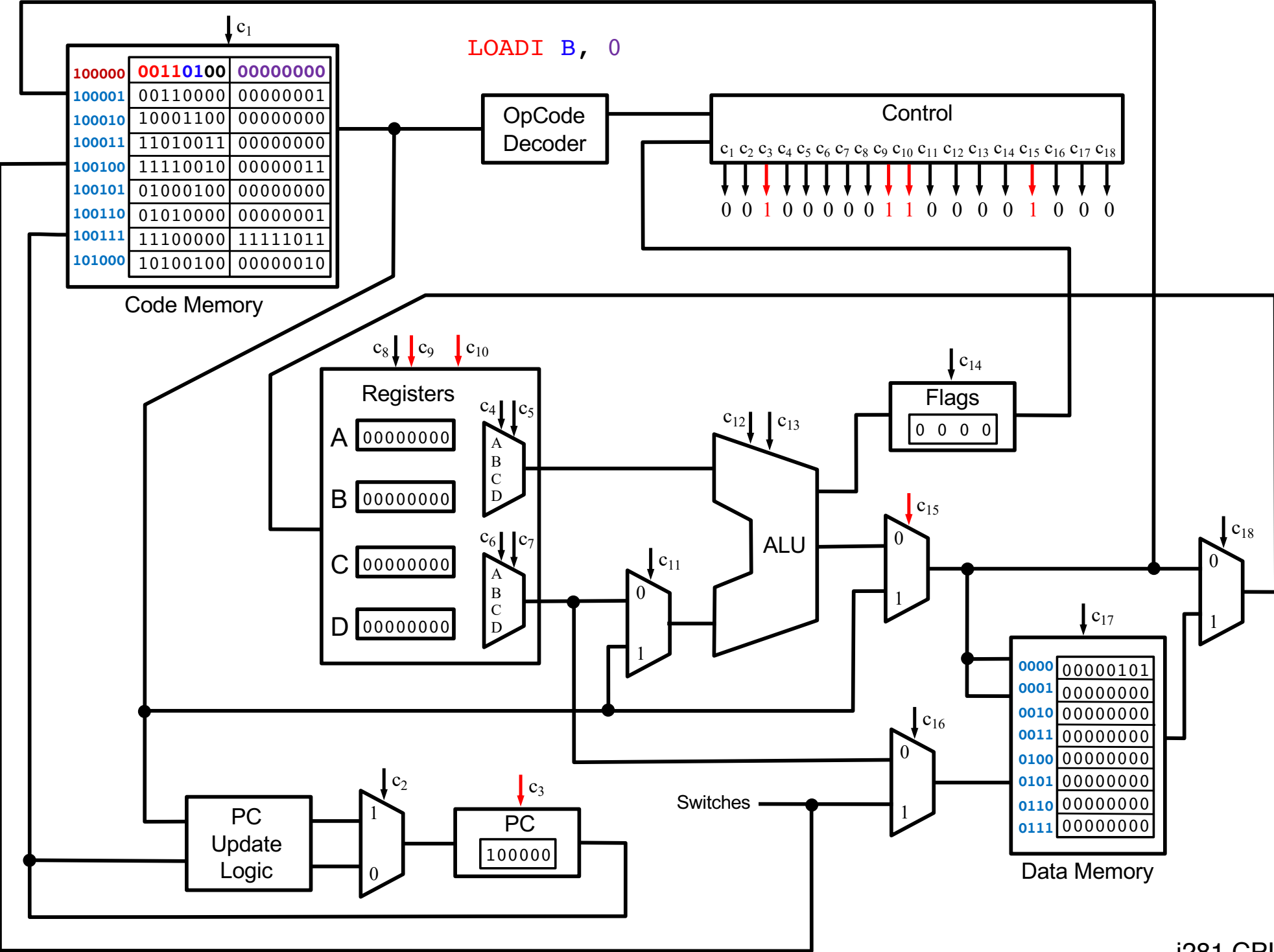
c<sub>2</sub> ↓

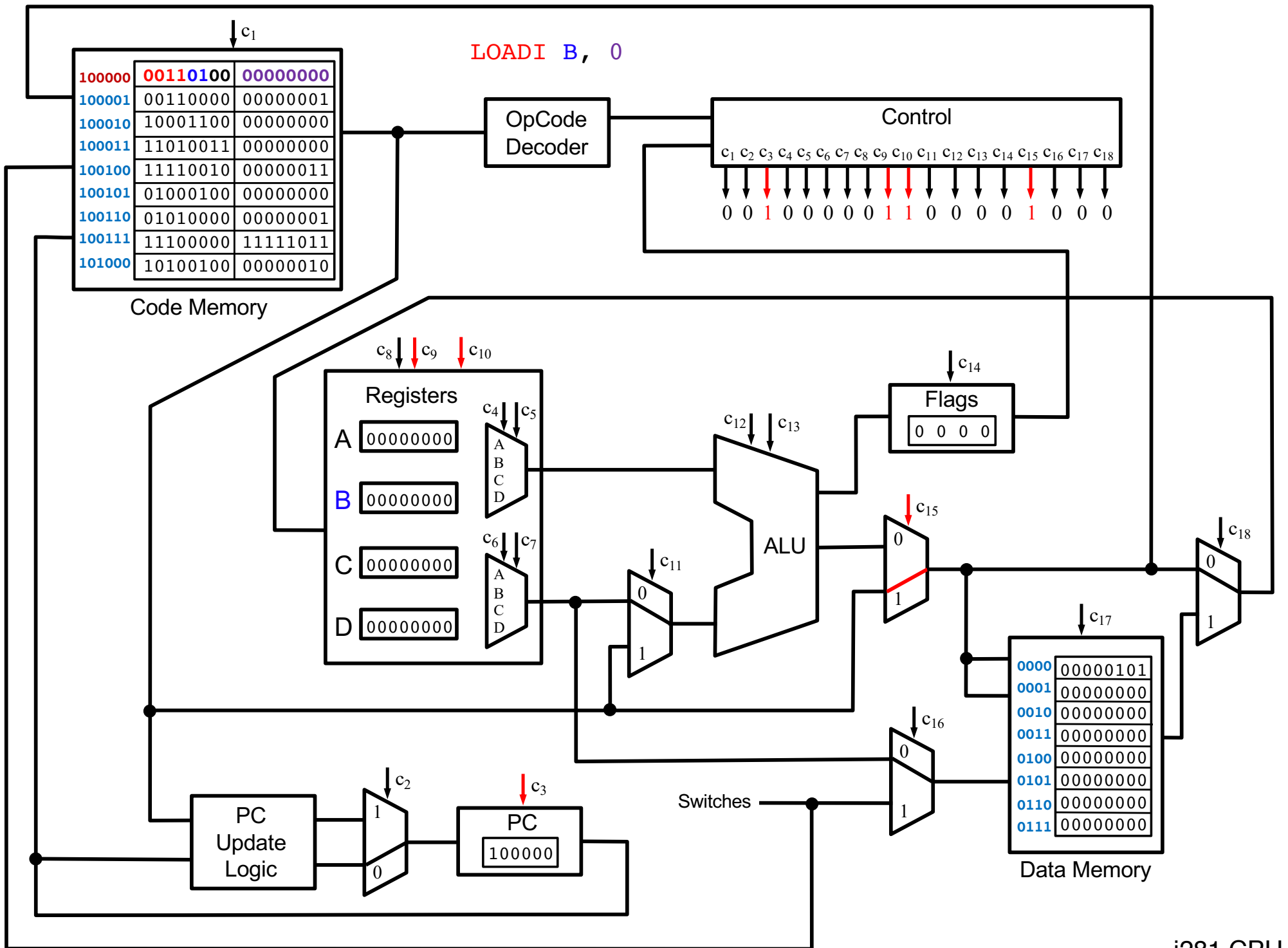
PC

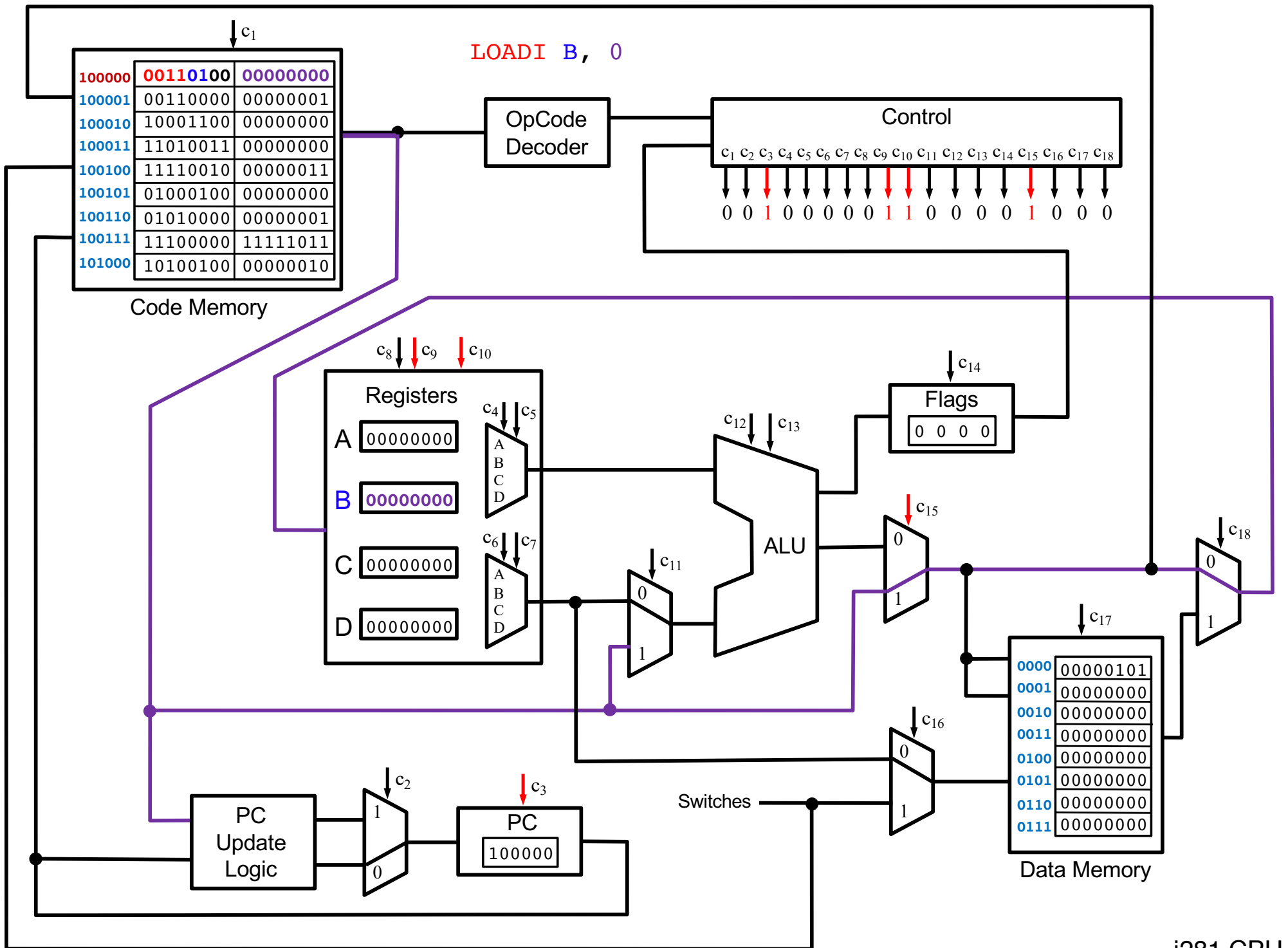
100000
--------

c<sub>3</sub> ↓

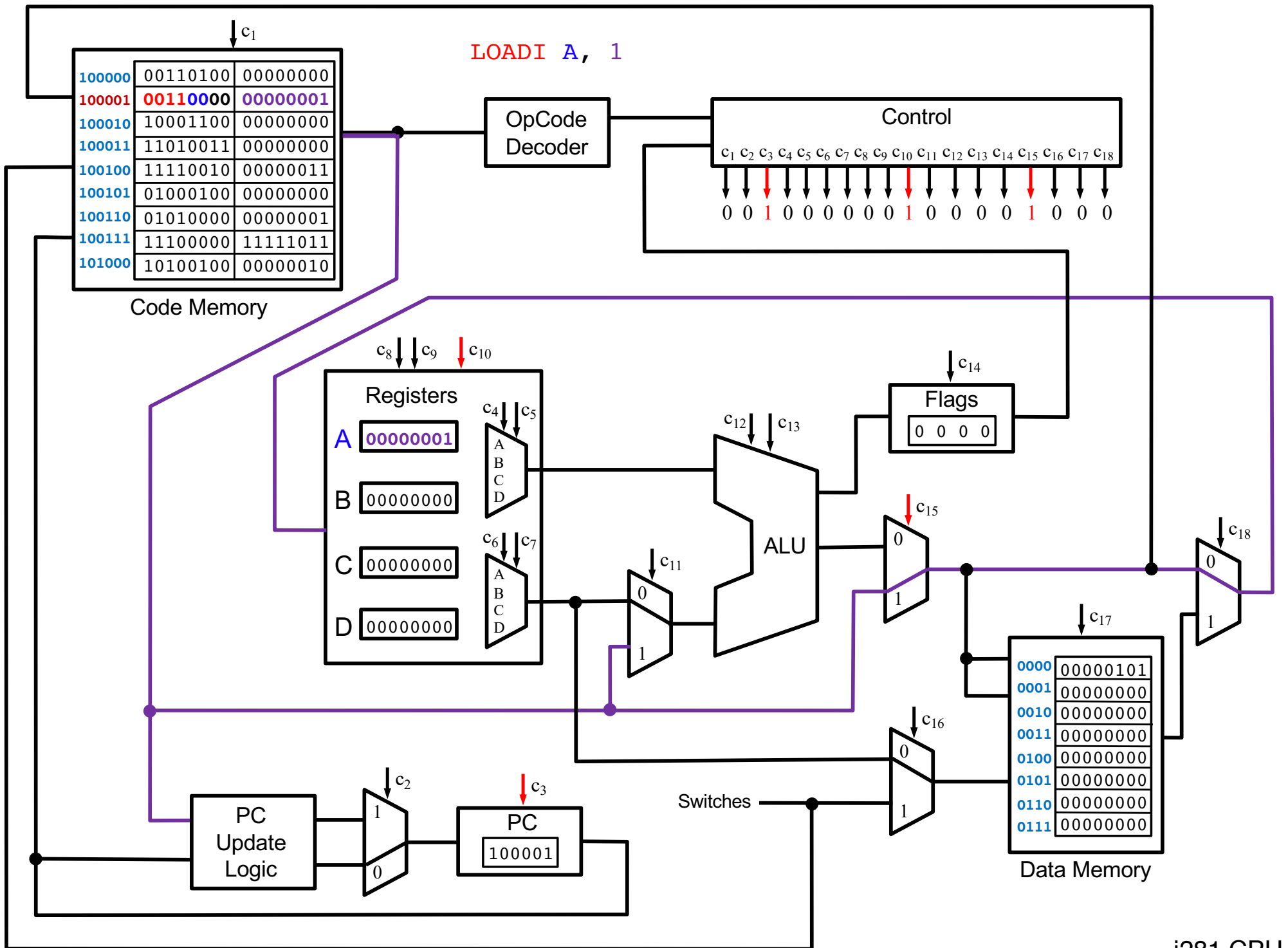
Switches













LOAD D, [N]

100000	00110100	00000000
100001	00110000	00000001
100010	10001100	00000000
100011	11010011	00000000
100100	11110010	00000011
100101	01000100	00000000
100110	01010000	00000001
100111	11110000	11111011
101000	10100100	00000010

Code Memory

OpCode Decoder

Control

$c_1$   $c_2$   $c_3$   $c_4$   $c_5$   $c_6$   $c_7$   $c_8$   $c_9$   $c_{10}$   $c_{11}$   $c_{12}$   $c_{13}$   $c_{14}$   $c_{15}$   $c_{16}$   $c_{17}$   $c_{18}$

0 0 1 0 0 0 0 1 1 1 0 0 0 0 1 0 0 1

Registers

A 00000001

B 00000000

C 00000000

D 00000000

$c_4$   $c_5$

A B C D

$c_6$   $c_7$

A B C D

ALU

$c_{12}$   $c_{13}$

Flags

0 0 0 0

$c_{14}$

Data Memory

0000	00000101
0001	00000000
0010	00000000
0011	00000000
0100	00000000
0101	00000000
0110	00000000
0111	00000000

$c_{17}$

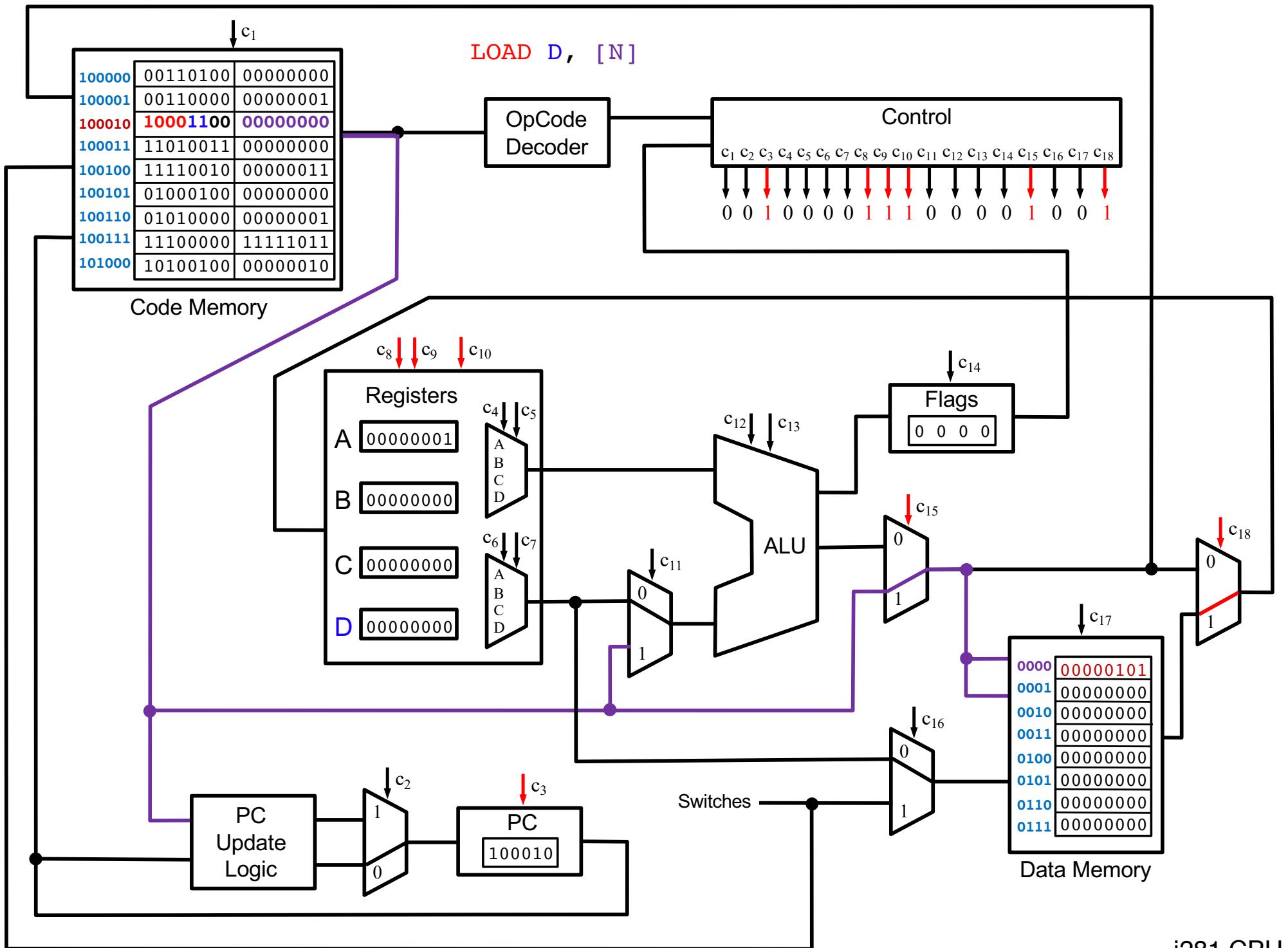
PC Update Logic

PC

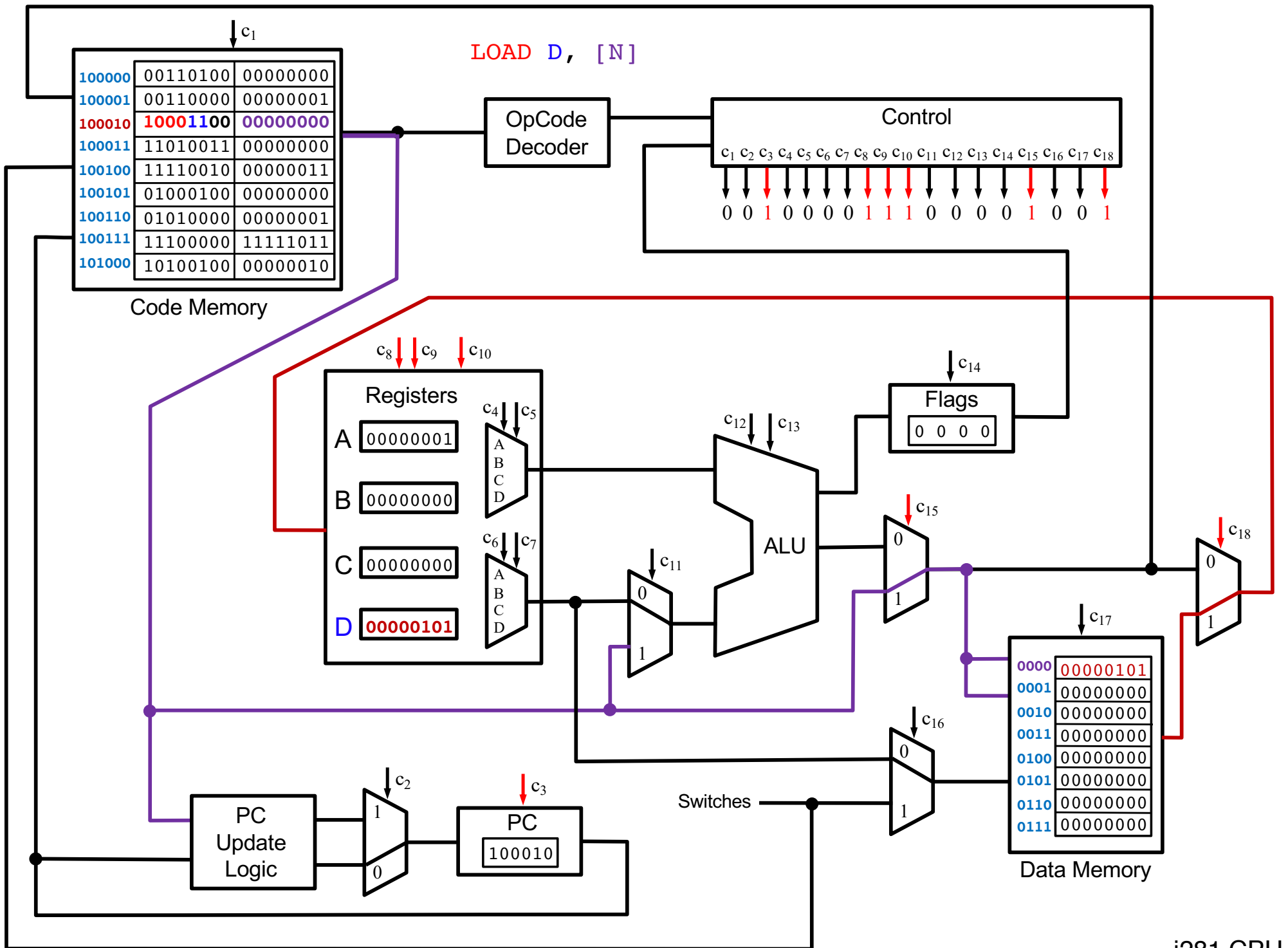
100010

$c_3$

Switches









CMP A, D

100000	00110100	00000000
100001	00110000	00000001
100010	10001100	00000000
100011	11010011	00000000
100100	11110010	00000011
100101	01000100	00000000
100110	01010000	00000001
100111	11100000	11111011
101000	10100100	00000010

Code Memory

OpCode Decoder

Control

c <sub>1</sub>	c <sub>2</sub>	c <sub>3</sub>	c <sub>4</sub>	c <sub>5</sub>	c <sub>6</sub>	c <sub>7</sub>	c <sub>8</sub>	c <sub>9</sub>	c <sub>10</sub>	c <sub>11</sub>	c <sub>12</sub>	c <sub>13</sub>	c <sub>14</sub>	c <sub>15</sub>	c <sub>16</sub>	c <sub>17</sub>	c <sub>18</sub>
0	0	1	0	0	1	1	0	0	0	0	1	1	1	0	0	0	0

Registers

A	00000001
B	00000000
C	00000000
D	00000101

Multiplexers for Register Selection:  
 Register A: c<sub>4</sub> (A), c<sub>5</sub> (B)  
 Register D: c<sub>6</sub> (A), c<sub>7</sub> (B)

ALU

Inputs: Register A (00000001), Register D (00000101)  
 Operation: CMP (Control bits c<sub>12</sub>=1, c<sub>13</sub>=1)  
 Output: 00000000

Flags

Control bit c<sub>14</sub> (Z) is 1.  
 Flag values: 0 0 0 0

PC Update Logic

Control bit c<sub>2</sub> is 0, selecting the ALU result (00000000) to update the PC.

PC

Control bit c<sub>3</sub> is 1, selecting the PC Update Logic output (00000000).  
 PC value: 100011

Switches

Data Memory

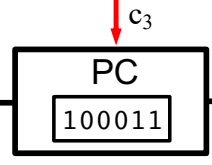
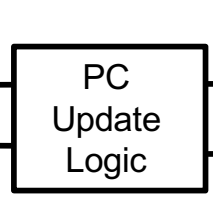
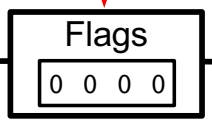
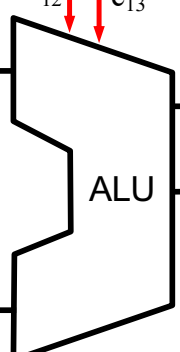
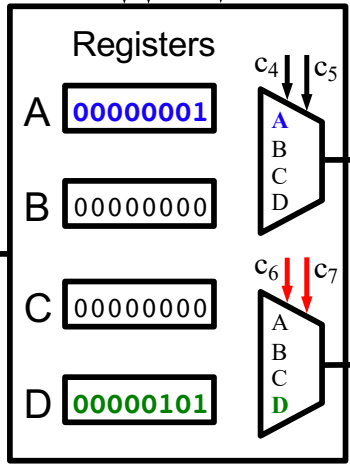
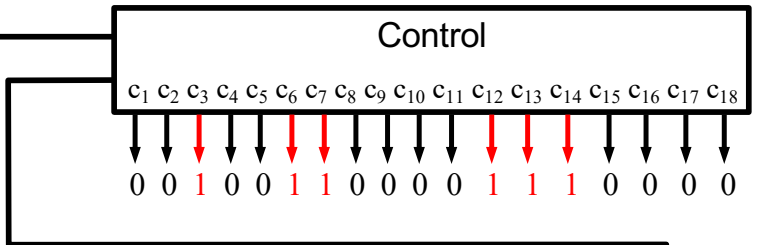
0000	00000101
0001	00000000
0010	00000000
0011	00000000
0100	00000000
0101	00000000
0110	00000000
0111	00000000

CMP A, D

100000	00110100	00000000
100001	00110000	00000001
100010	10001100	00000000
100011	11010011	00000000
100100	11110010	00000011
100101	01000100	00000000
100110	01010000	00000001
100111	11110000	11111011
101000	10100100	00000010

Code Memory

OpCode Decoder



Switches

Data Memory

0000	00000101
0001	00000000
0010	00000000
0011	00000000
0100	00000000
0101	00000000
0110	00000000
0111	00000000

Input: c<sub>17</sub>

CMP A, D

100000	00110100	00000000
100001	00110000	00000001
100010	10001100	00000000
100011	11010011	00000000
100100	11110010	00000011
100101	01000100	00000000
100110	01010000	00000001
100111	11110000	11111011
101000	10100100	00000010

Code Memory

OpCode Decoder

Control

c <sub>1</sub>	c <sub>2</sub>	c <sub>3</sub>	c <sub>4</sub>	c <sub>5</sub>	c <sub>6</sub>	c <sub>7</sub>	c <sub>8</sub>	c <sub>9</sub>	c <sub>10</sub>	c <sub>11</sub>	c <sub>12</sub>	c <sub>13</sub>	c <sub>14</sub>	c <sub>15</sub>	c <sub>16</sub>	c <sub>17</sub>	c <sub>18</sub>
0	0	1	0	0	1	1	0	0	0	0	1	1	1	0	0	0	0

Registers

A	00000001
B	00000000
C	00000000
D	00000101

Multiplexers for Register Selection:  
 A: c<sub>4</sub>, c<sub>5</sub>  
 B: c<sub>4</sub>, c<sub>5</sub>  
 C: c<sub>6</sub>, c<sub>7</sub>  
 D: c<sub>6</sub>, c<sub>7</sub>

ALU

Inputs: 00000001 (A), 00000101 (D)

Carry-ins: c<sub>12</sub>, c<sub>13</sub>

Flags

Output: 0 0 0 0

Carry-in: c<sub>14</sub>

Multiplexer

Carry-in: c<sub>11</sub>

Output: 0

Multiplexer

Carry-in: c<sub>15</sub>

Output: 0

Multiplexer

Carry-in: c<sub>18</sub>

Output: 0

PC Update Logic

Multiplexer

Carry-in: c<sub>2</sub>

Output: 1

PC

Value: 100011

Carry-in: c<sub>3</sub>

Switches

Multiplexer

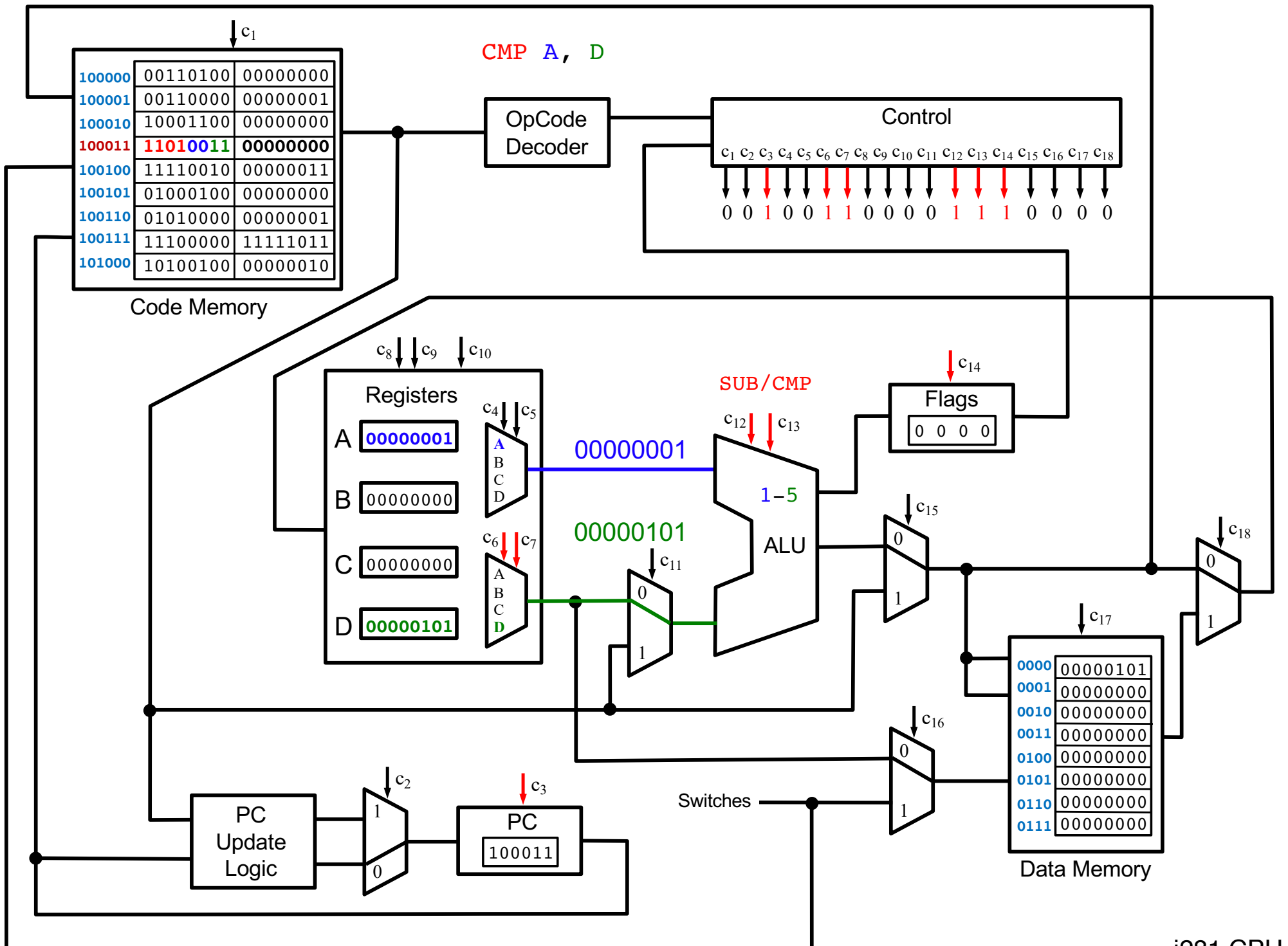
Carry-in: c<sub>16</sub>

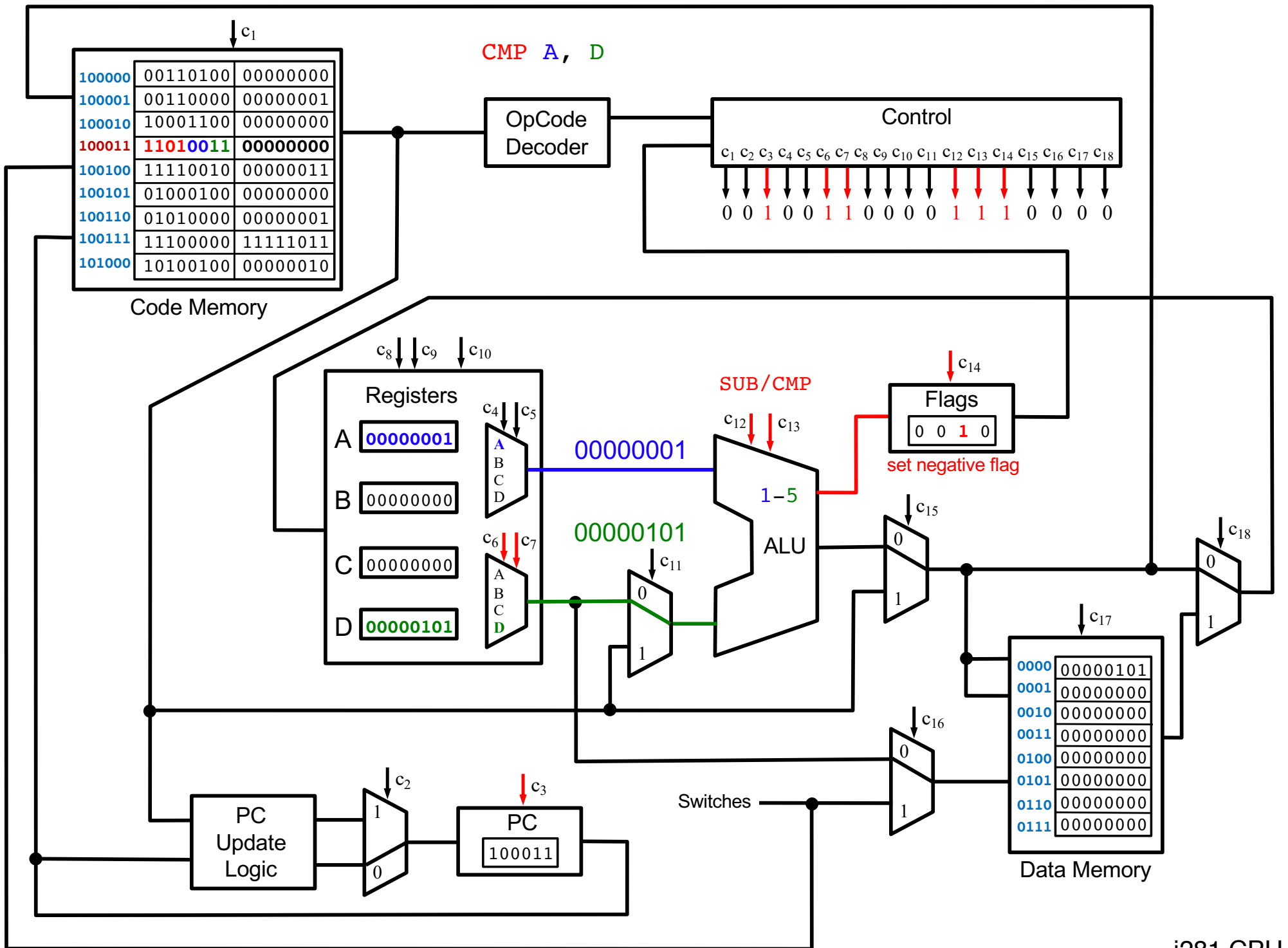
Output: 0

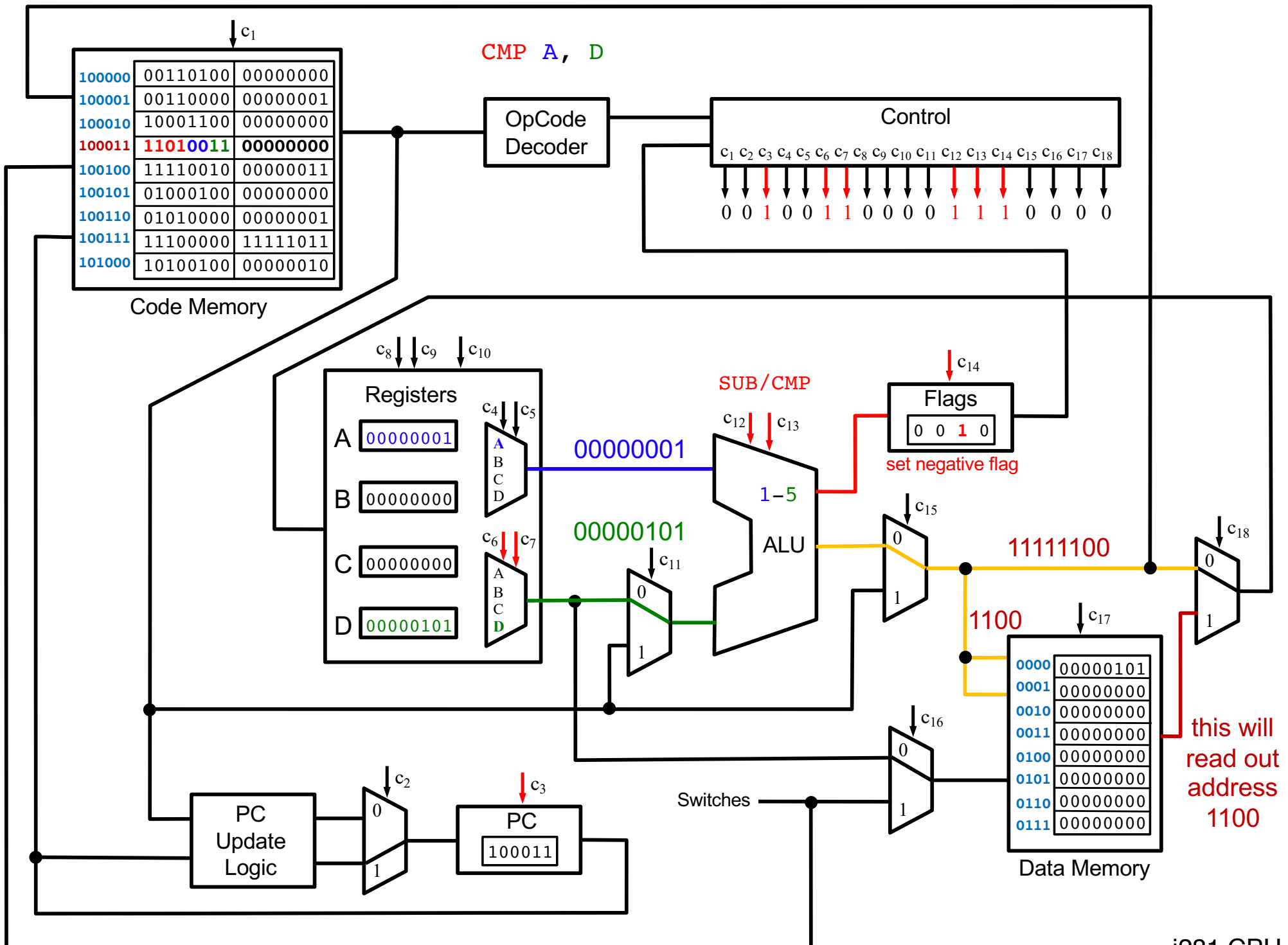
Data Memory

0000	00000101
0001	00000000
0010	00000000
0011	00000000
0100	00000000
0101	00000000
0110	00000000
0111	00000000

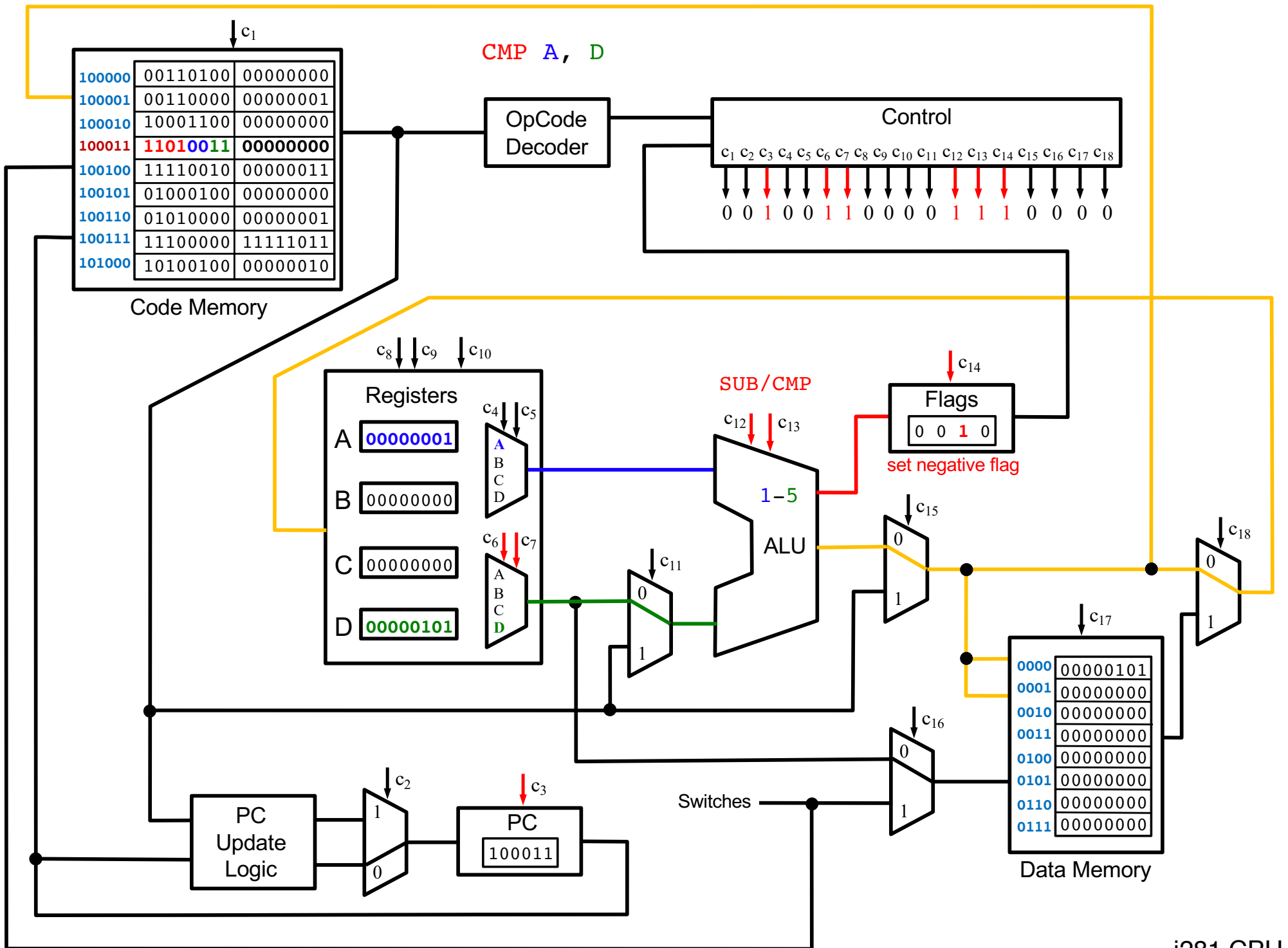
Carry-in: c<sub>17</sub>

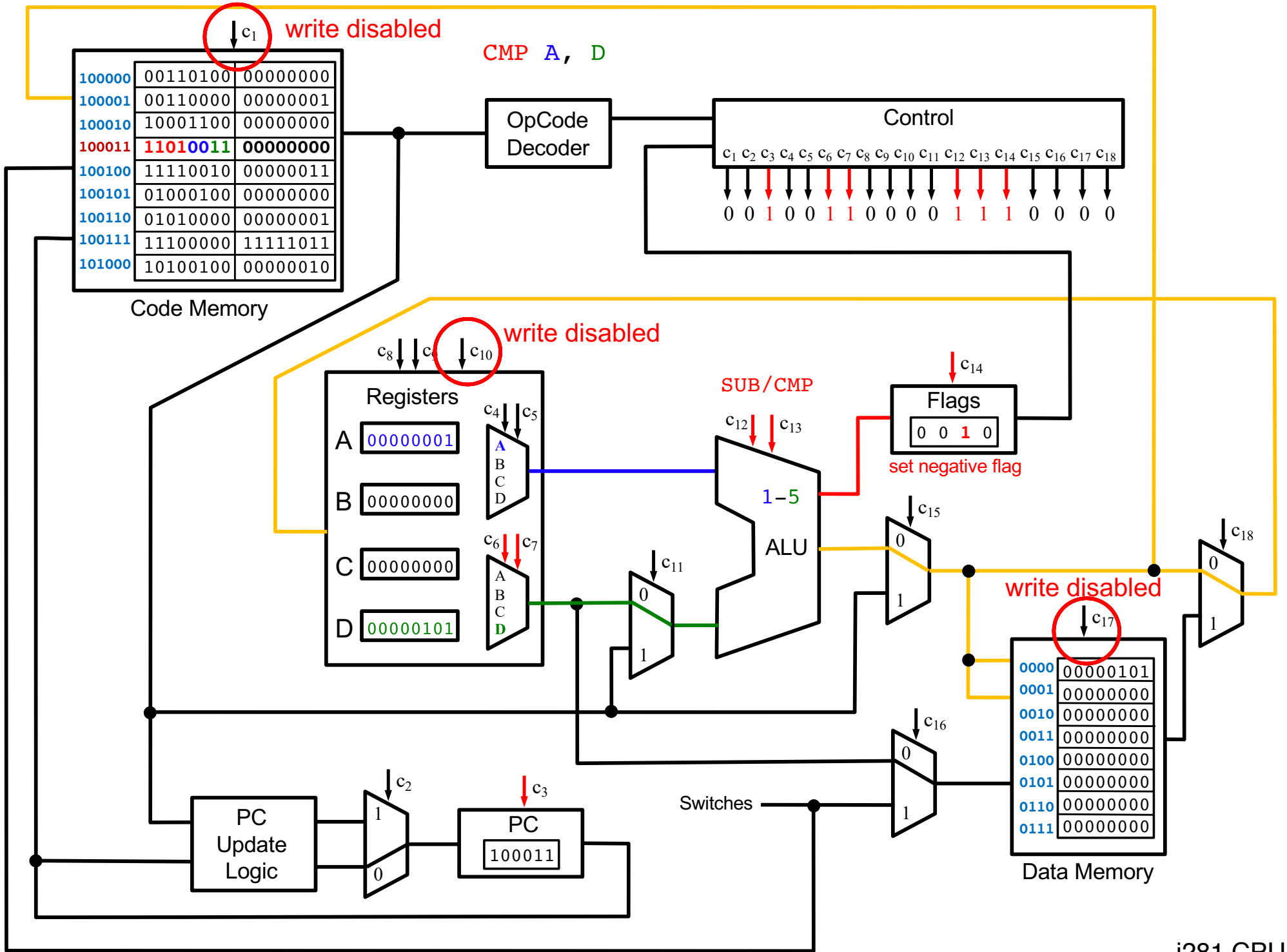


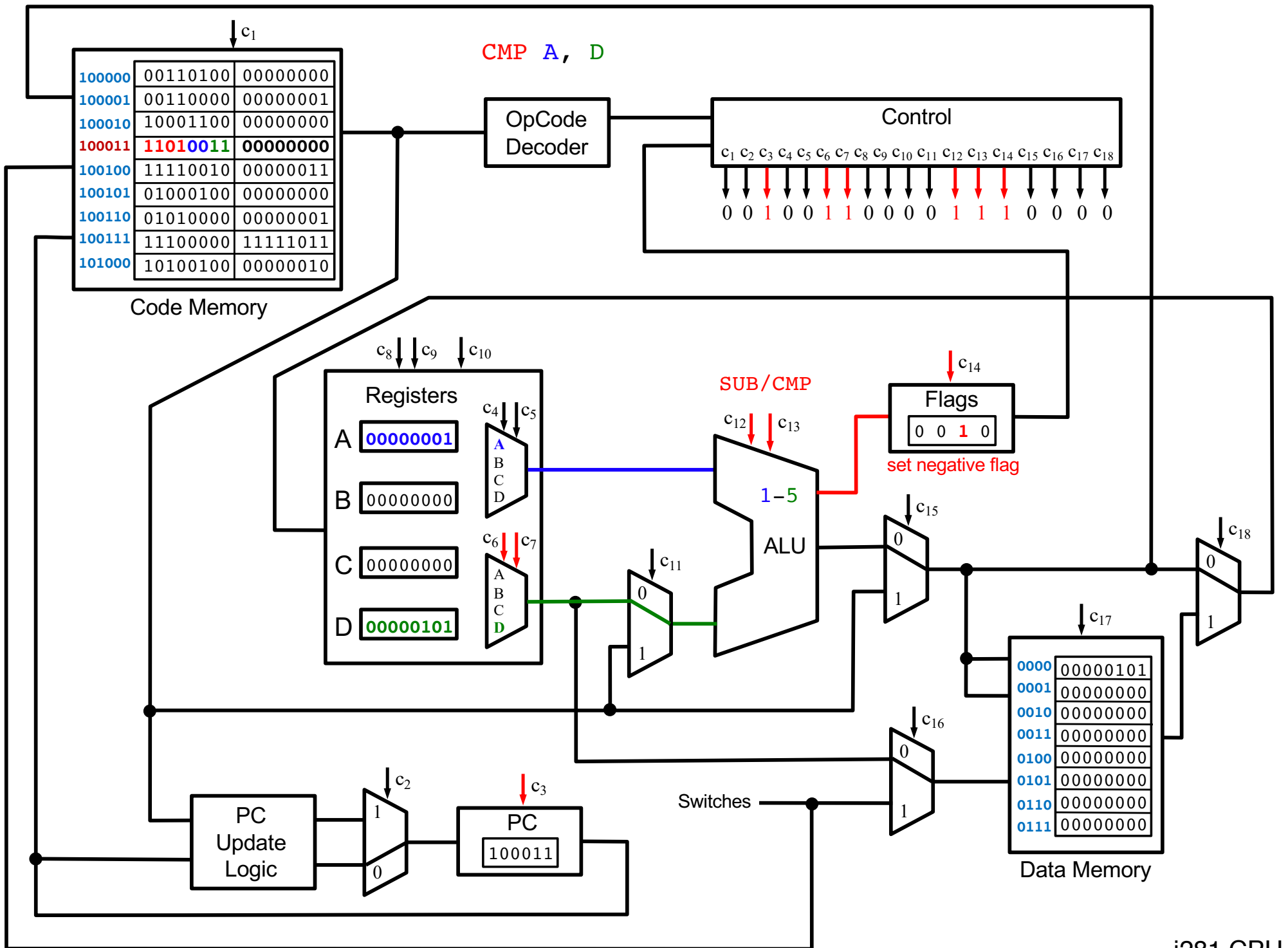












CMP A, D

100000	00110100	00000000
100001	00110000	00000001
100010	10001100	00000000
100011	11010011	00000000
100100	11110010	00000011
100101	01000100	00000000
100110	01010000	00000001
100111	11110000	11111011
101000	10100100	00000010

Code Memory

OpCode Decoder

Control

c <sub>1</sub>	c <sub>2</sub>	c <sub>3</sub>	c <sub>4</sub>	c <sub>5</sub>	c <sub>6</sub>	c <sub>7</sub>	c <sub>8</sub>	c <sub>9</sub>	c <sub>10</sub>	c <sub>11</sub>	c <sub>12</sub>	c <sub>13</sub>	c <sub>14</sub>	c <sub>15</sub>	c <sub>16</sub>	c <sub>17</sub>	c <sub>18</sub>
0	0	1	0	0	1	1	0	0	0	0	1	1	1	0	0	0	0

Registers

A	00000001
B	00000000
C	00000000
D	00000101

ALU

Flags

0	0	1	0
---	---	---	---

The only effect of this operation

PC Update Logic

PC

100011
--------

Switches

Data Memory

0000	00000101
0001	00000000
0010	00000000
0011	00000000
0100	00000000
0101	00000000
0110	00000000
0111	00000000



BRG End

100000	00110100	00000000
100001	00110000	00000001
100010	10001100	00000000
100011	11010011	00000000
100100	11110010	00000011
100101	01000100	00000000
100110	01010000	00000001
100111	11110000	11111011
101000	10100100	00000010

Code Memory

OpCode Decoder

Control

c <sub>1</sub>	c <sub>2</sub>	c <sub>3</sub>	c <sub>4</sub>	c <sub>5</sub>	c <sub>6</sub>	c <sub>7</sub>	c <sub>8</sub>	c <sub>9</sub>	c <sub>10</sub>	c <sub>11</sub>	c <sub>12</sub>	c <sub>13</sub>	c <sub>14</sub>	c <sub>15</sub>	c <sub>16</sub>	c <sub>17</sub>	c <sub>18</sub>	
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Registers

A	00000001
B	00000000
C	00000000
D	00000101

c<sub>4</sub> ↓ c<sub>5</sub> ↓  
 A B C D  
 c<sub>6</sub> ↓ c<sub>7</sub> ↓  
 A B C D

ALU

c<sub>12</sub> ↓ c<sub>13</sub> ↓

Flags

0	0	1	0
---	---	---	---

c<sub>14</sub> ↓

Data Memory

0000	00000101
0001	00000000
0010	00000000
0011	00000000
0100	00000000
0101	00000000
0110	00000000
0111	00000000

c<sub>17</sub> ↓

PC Update Logic

c<sub>2</sub> ↓

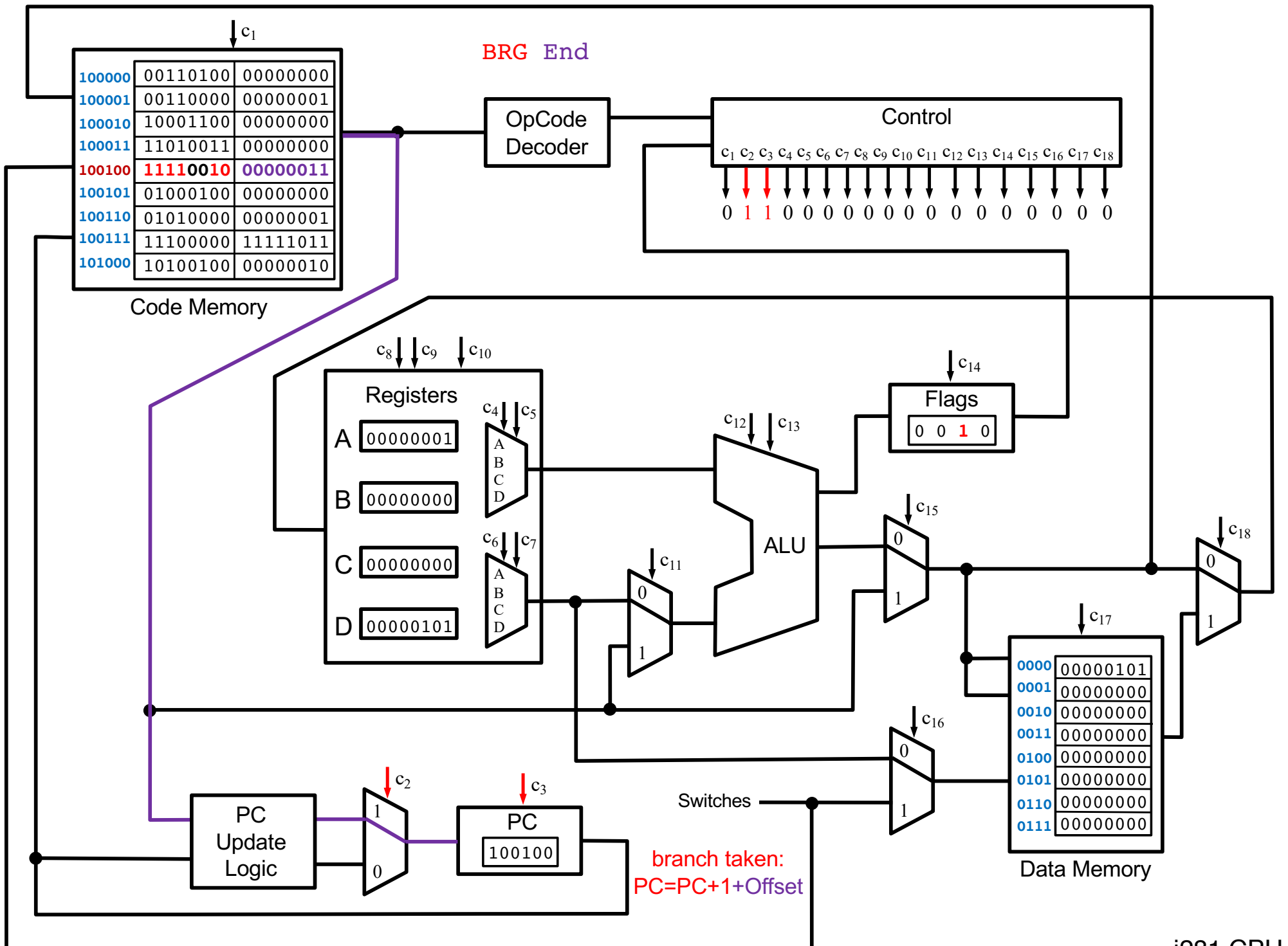
PC

100100
--------

c<sub>3</sub> ↓

branch not taken: PC=PC+1

Switches



**Questions?**



**THE END**