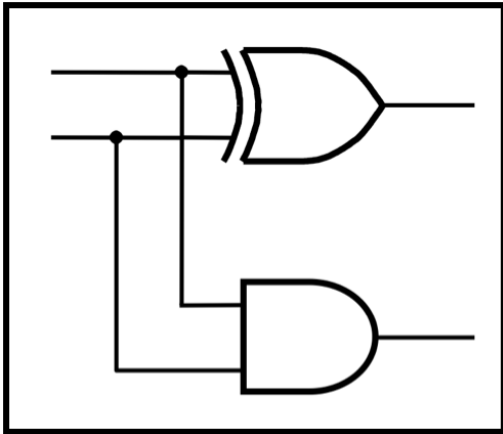




CprE 2810: Digital Logic

Instructor: Alexander Stoytchev

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

The Intersection Between Hardware and Software

*CprE 2810: Digital Logic
Iowa State University, Ames, IA
Copyright © Alexander Stoytchev*

Administrative Stuff

- The FINAL exam is scheduled for
- **Wednesday** Dec 17 @ 2:15 – 4:15 PM

Final Exam Format

- **The exam will cover: Chapter 1 to Chapter 6, and Sections 7.1-7.2, register machines, and i281 CPU**
- **Emphasis will be on Chapter 5, 6, and 7**
- **The exam will be closed book but open notes.**
- **You can bring up to 5 sheets of handwritten or typed notes.**

Final Exam Format

- The exam will be out of 135 points
- You need 95 points to get an A on this exam
- It will be great if you can score more than 100 points.
 - but you can't roll over your extra points ☹

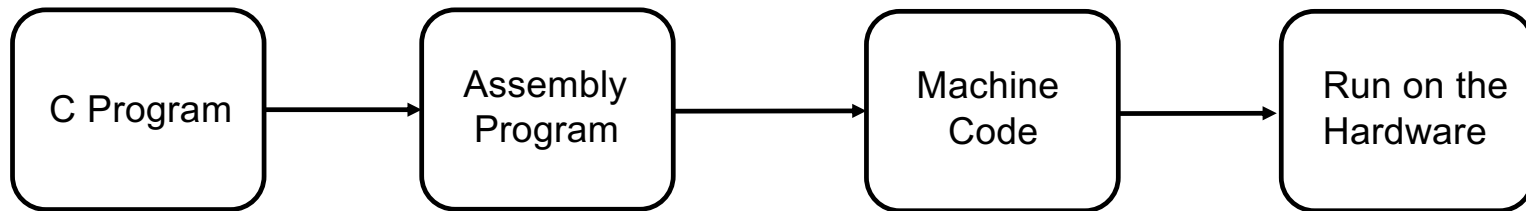
Topics for the Final Exam

- **K-maps for 2, 3, and 4 variables**
- **Multiplexers (circuits and function)**
- **Synthesis of logic functions using multiplexers**
- **Shannon's Expansion Theorem**
- **1's complement and 2's complement representation**
- **Addition and subtraction of binary numbers**
- **Circuits for adding and subtracting**
- **Serial adder**
- **Latches (circuits, behavior, timing diagrams)**
- **Flip-Flops (circuits, behavior, timing diagrams)**
- **Counters (up, down, synchronous, asynchronous)**
- **Registers and Register Files**

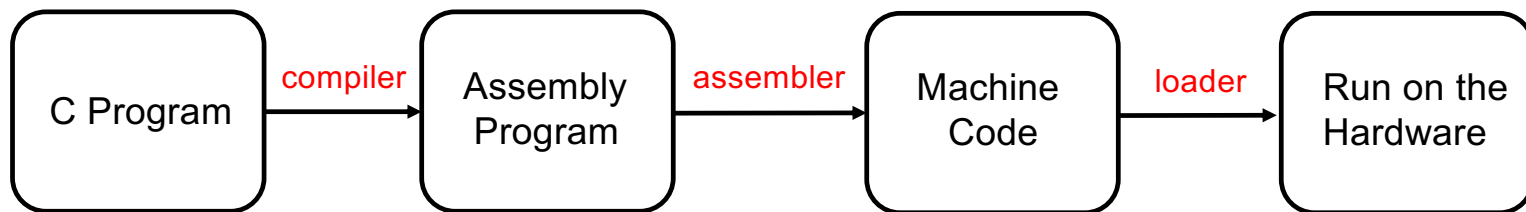
Topics for the Final Exam

- Synchronous Sequential Circuits
- FSMs
- Moore Machines
- Mealy Machines
- State diagrams, state tables, state-assigned tables
- State minimization
- Designing a counter
- Arbiter Circuits
- Reverse engineering a circuit
- ASM Charts
- Register Machines and programs for them
- ALU, PC, and control for a simple processor (i281 CPU)
- Assembly and machine language (i281 assembly)
- Something from Star Wars

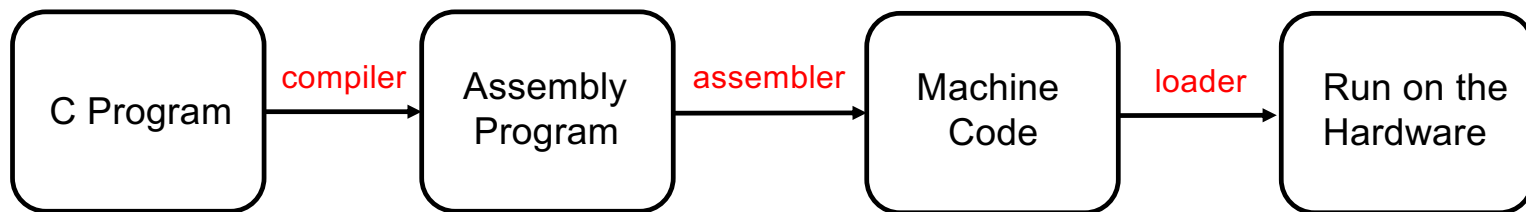
Writing and Running a Program



Writing and Running a Program

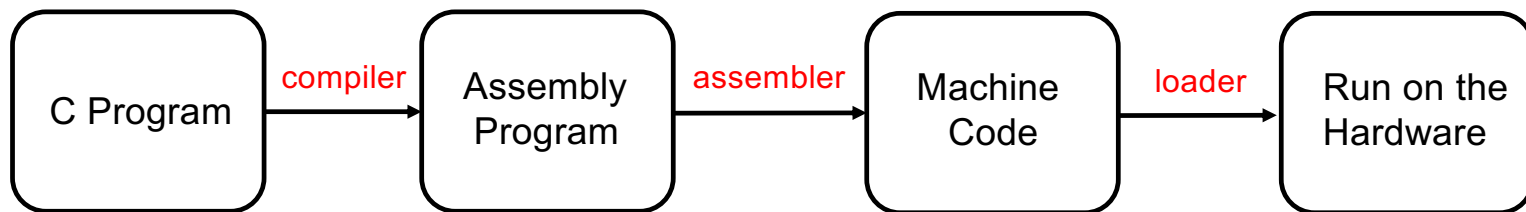


Writing and Running a Program



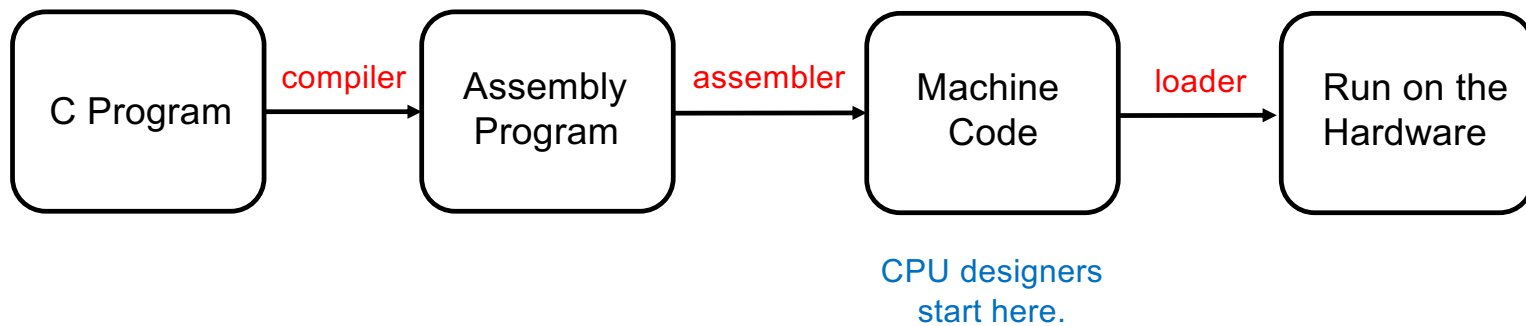
The programmer
only writes this
in a text editor.

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

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

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

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

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

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

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

“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

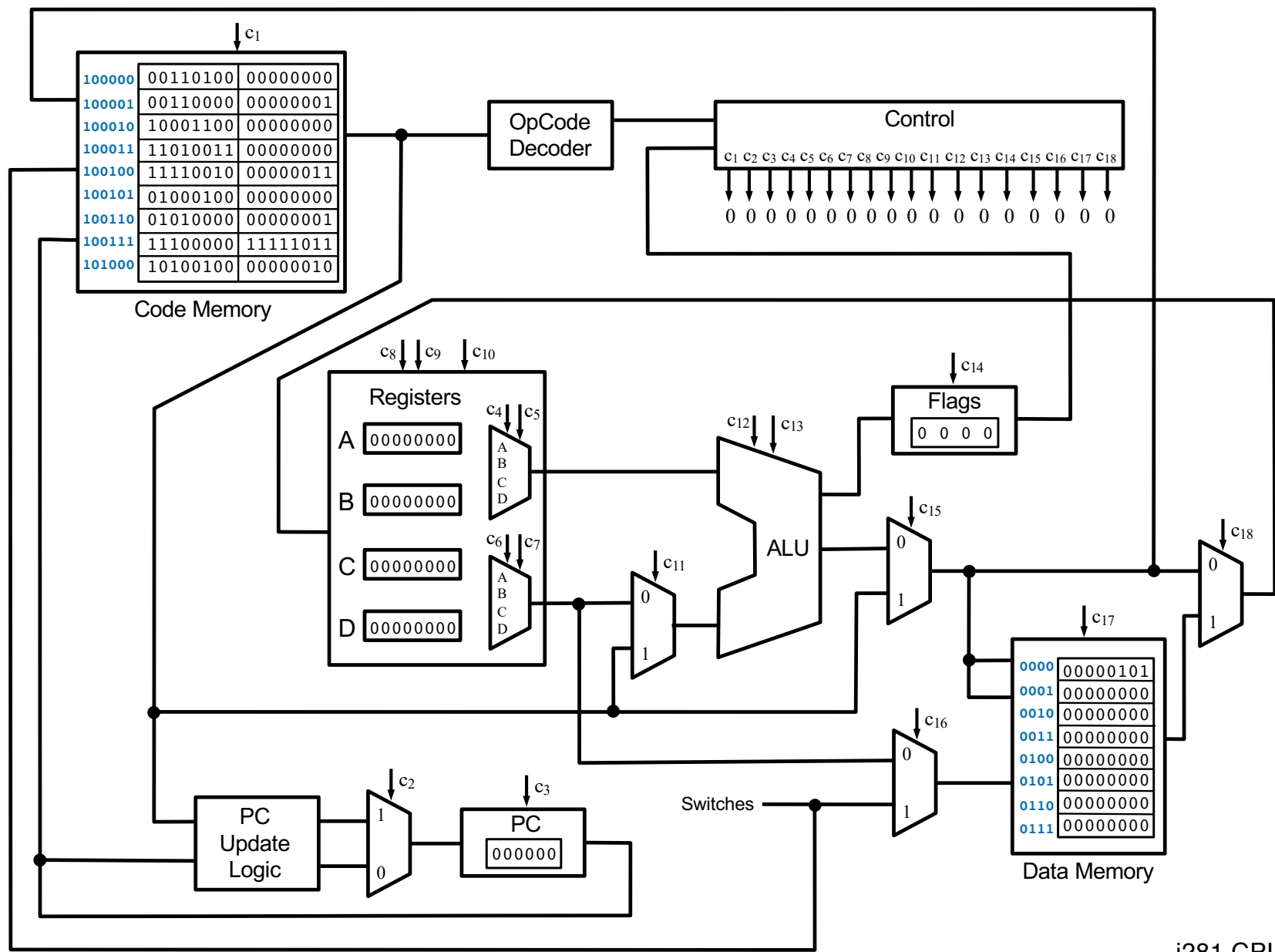
.code

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**

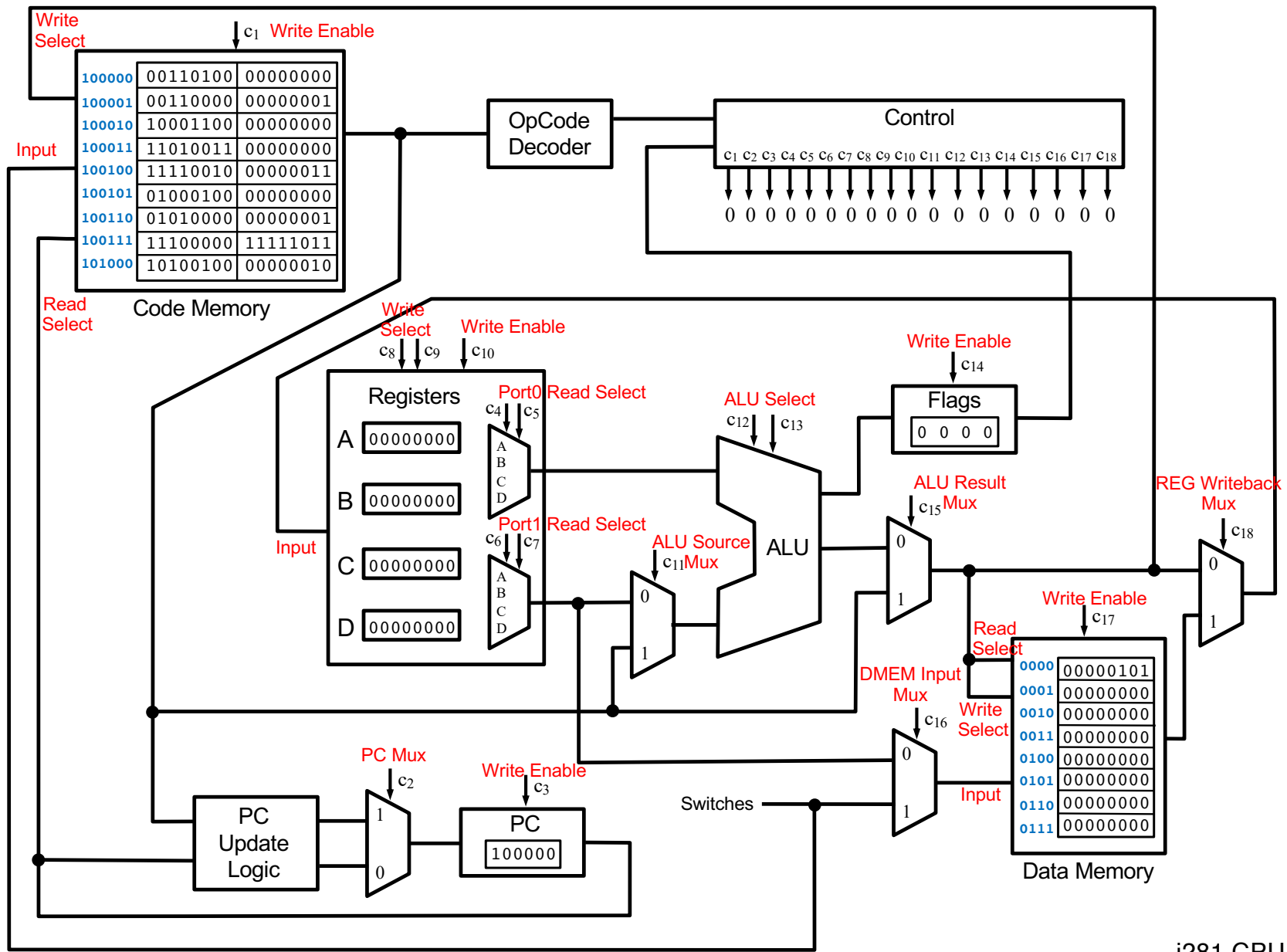
Code Memory:

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

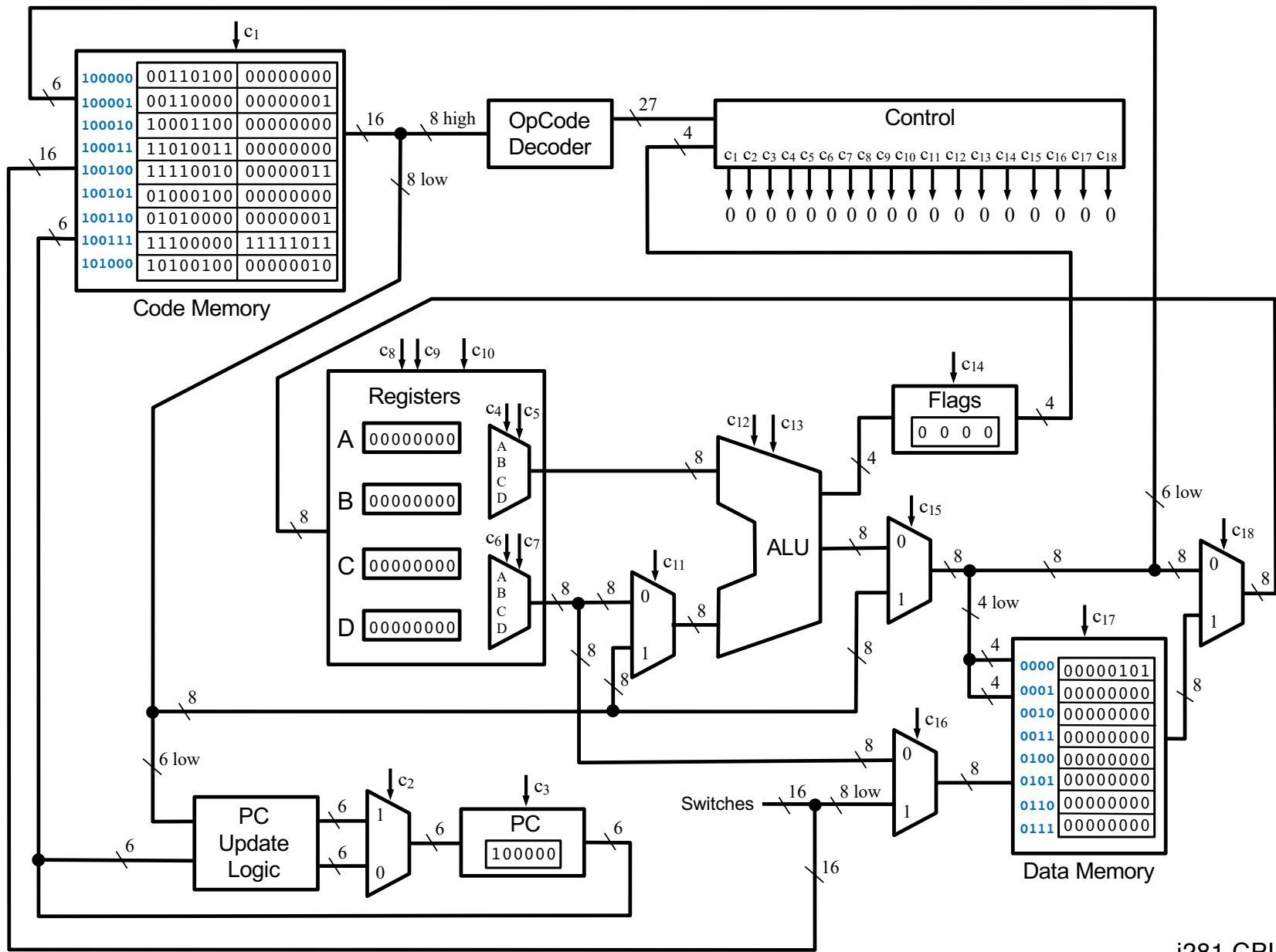
Loading the Program into Memory



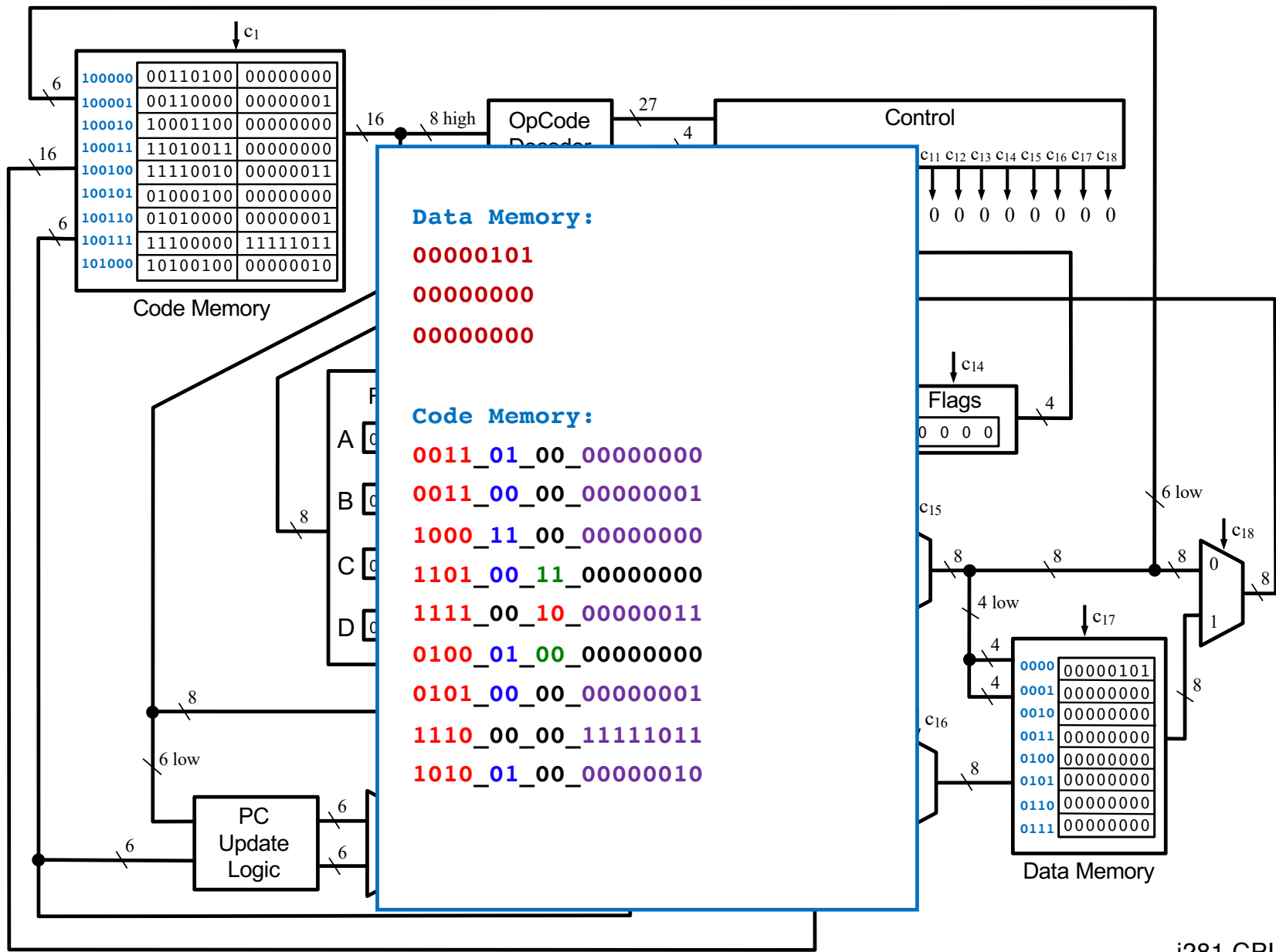
i281 CPU



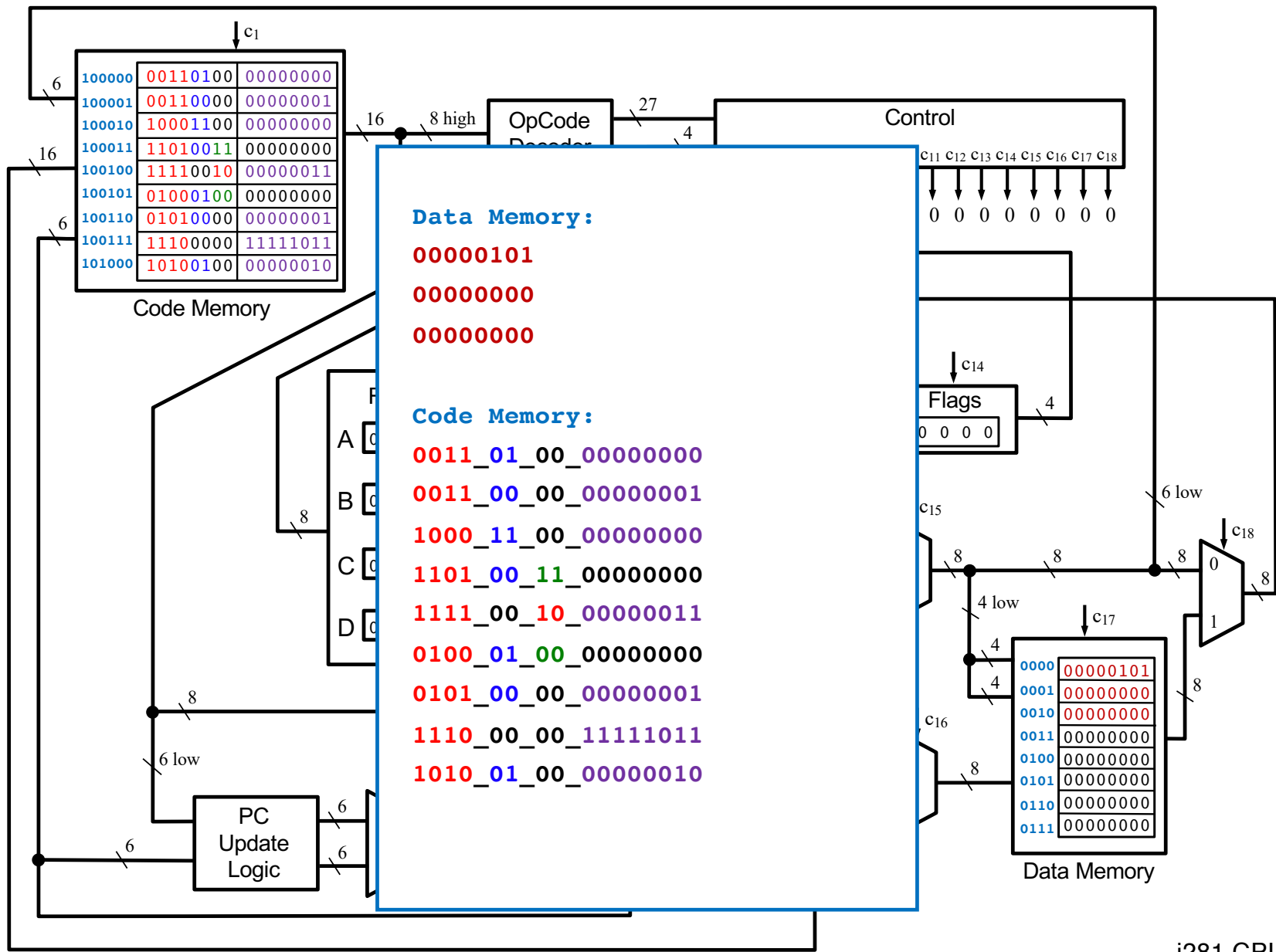
i281 CPU



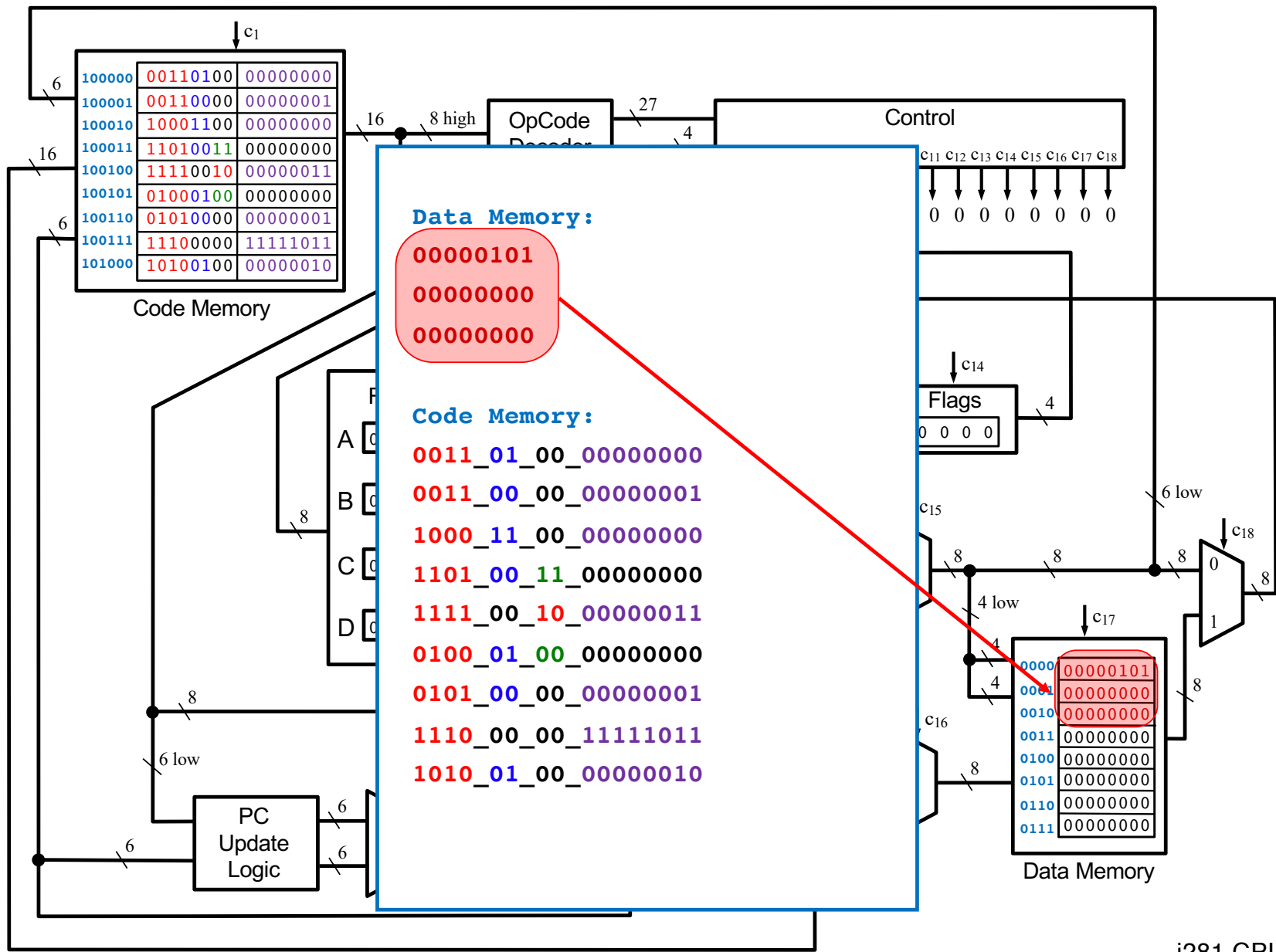
i281 CPU



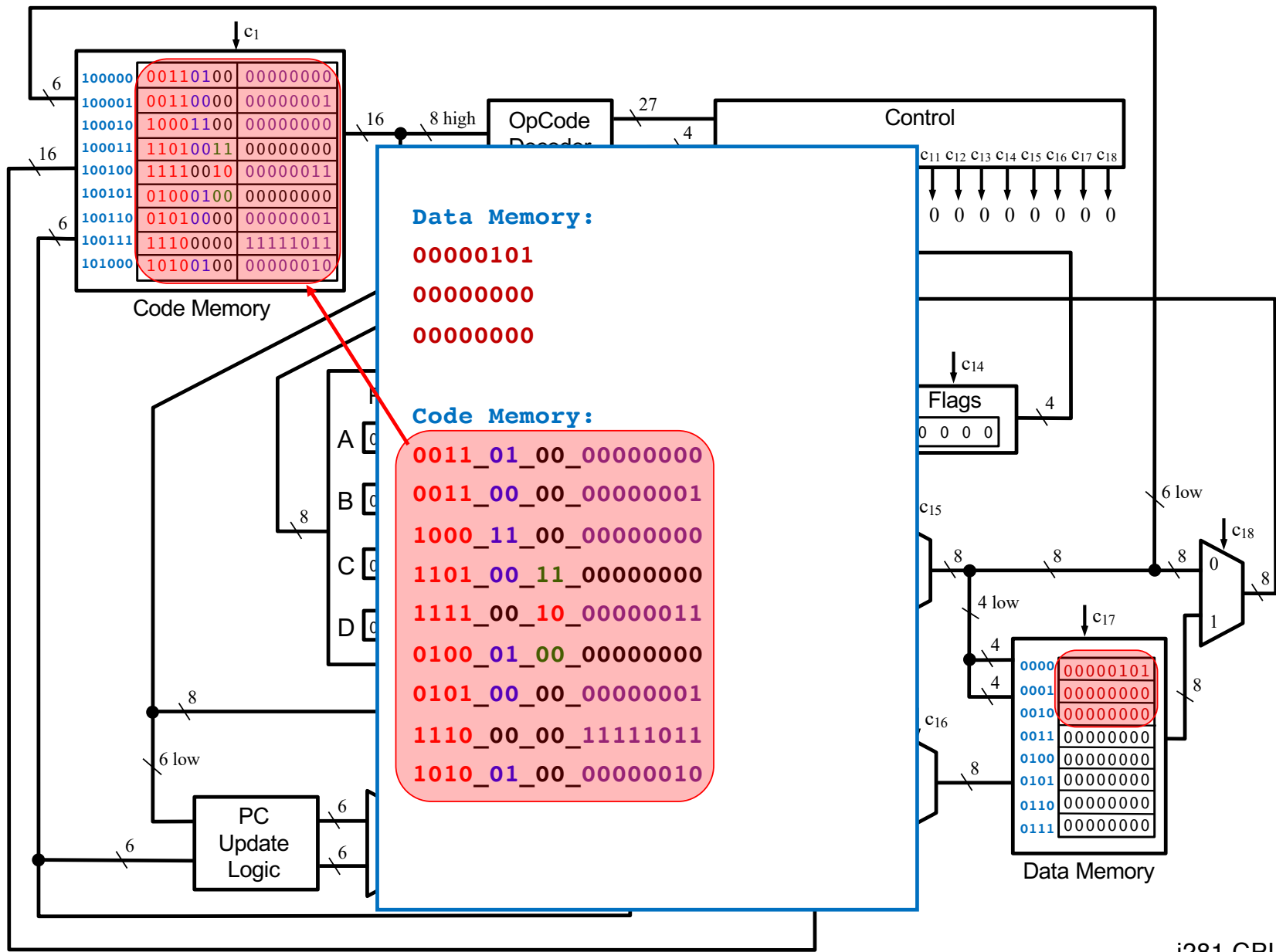
i281 CPU



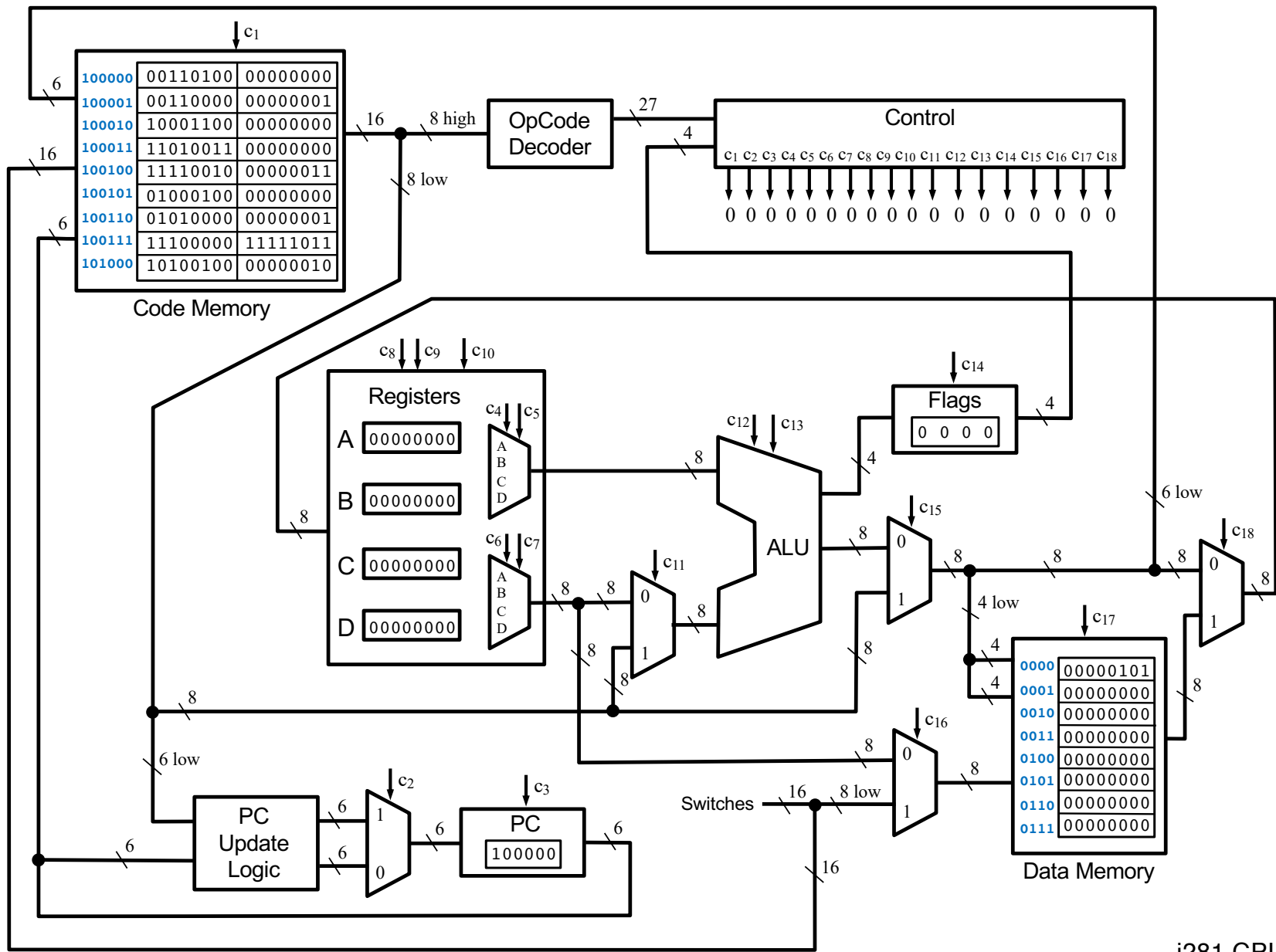
i281 CPU



i281 CPU

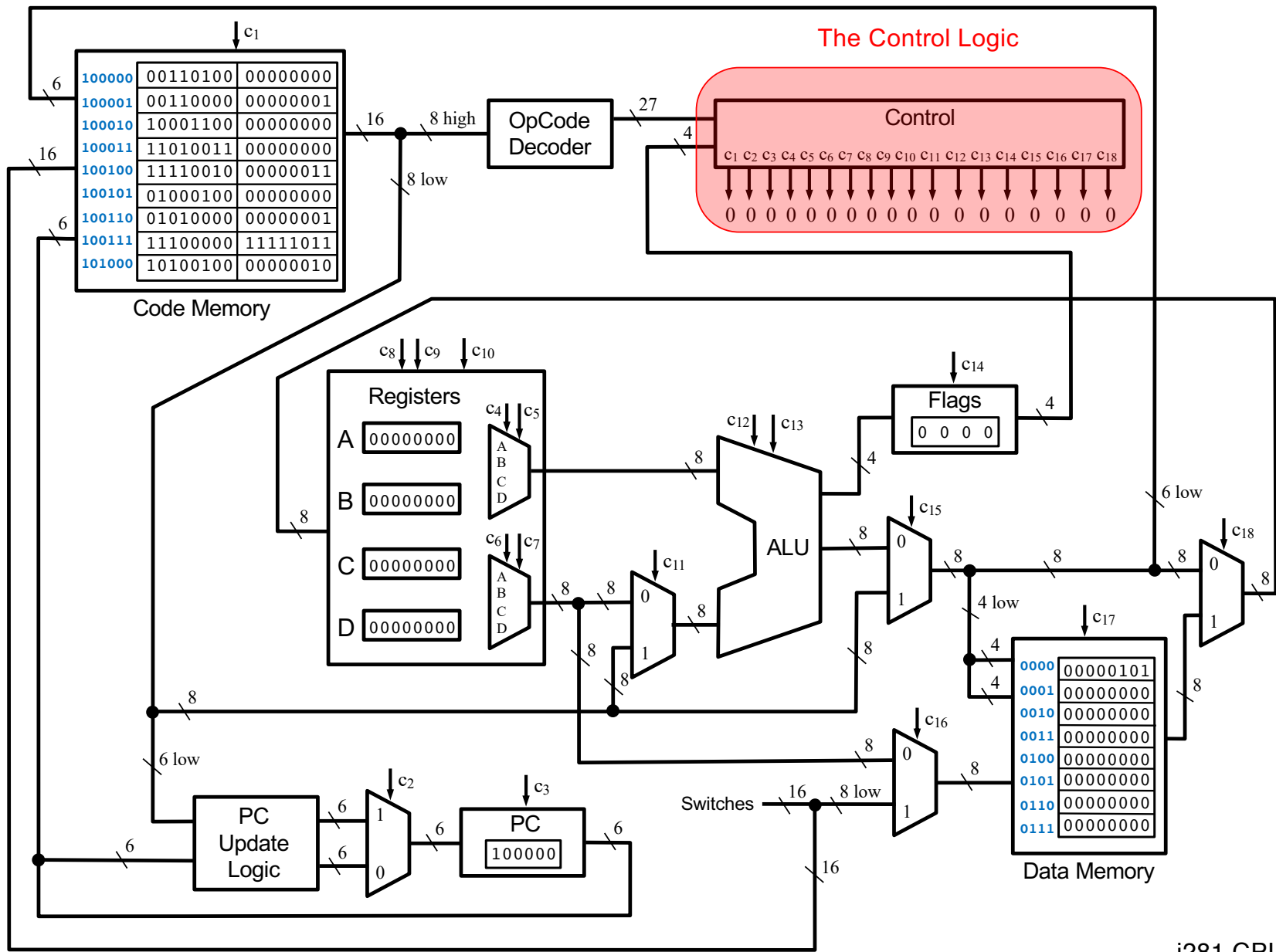


i281 CPU

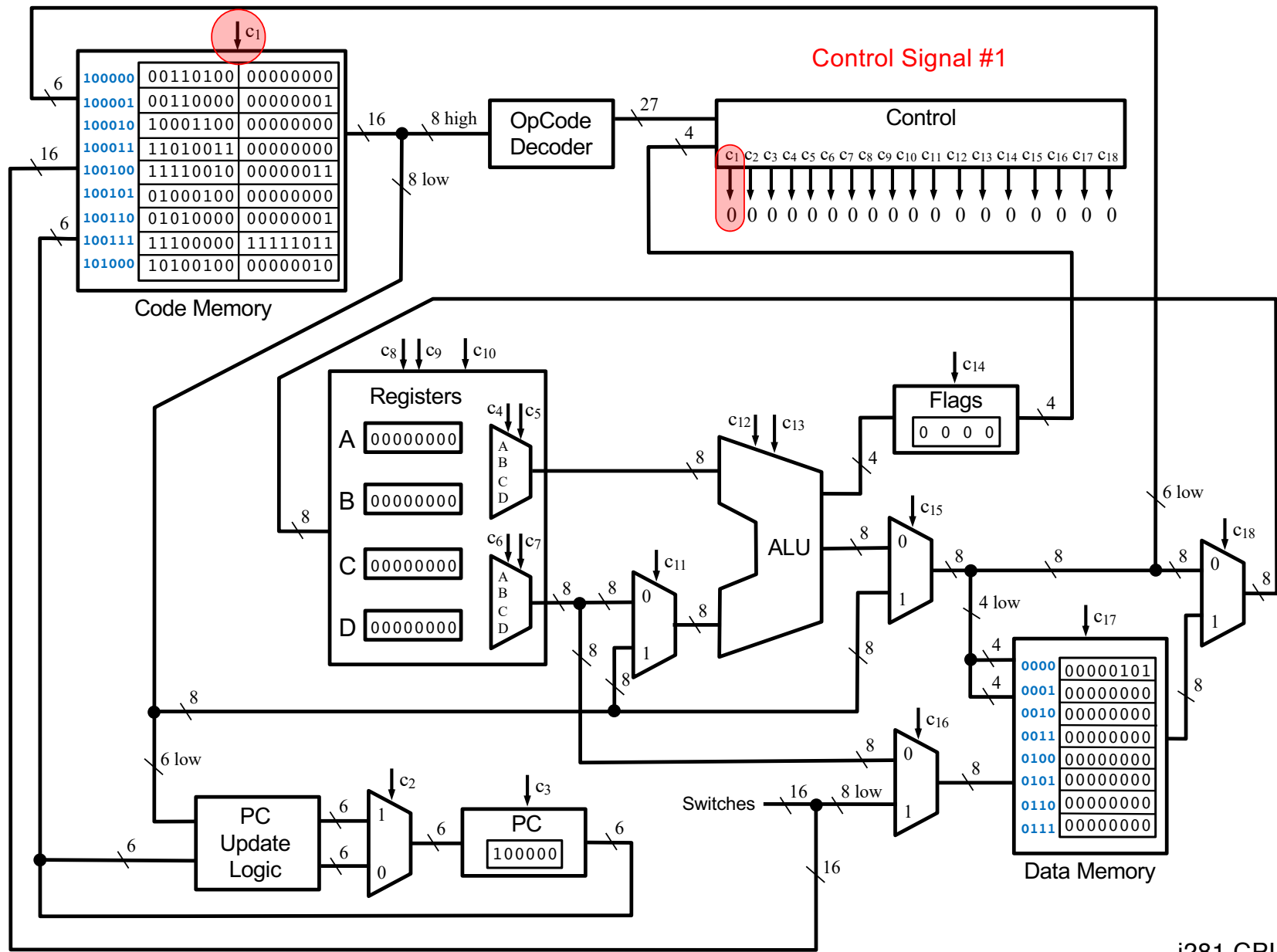


i281 CPU

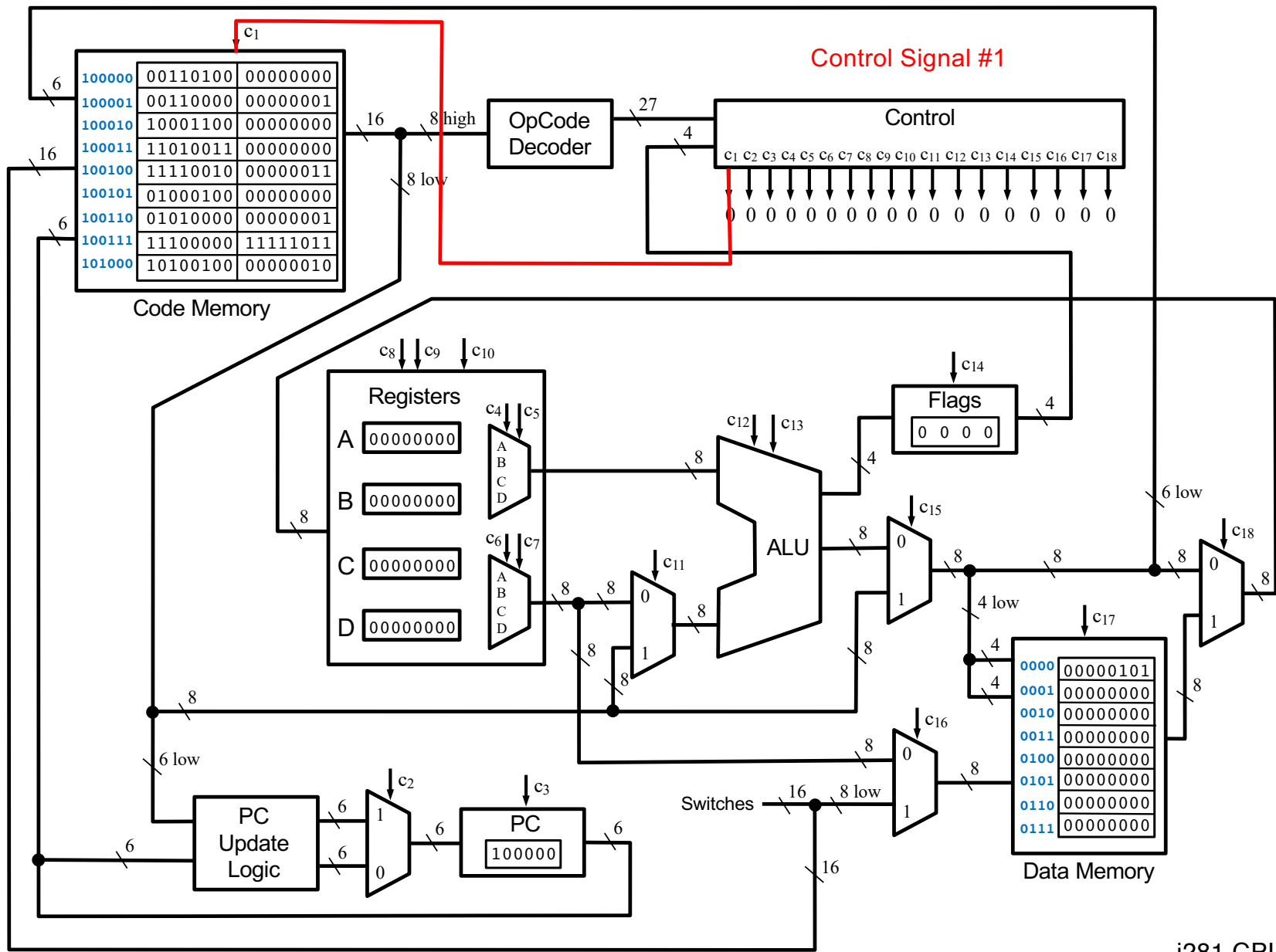
The CPU Control Logic



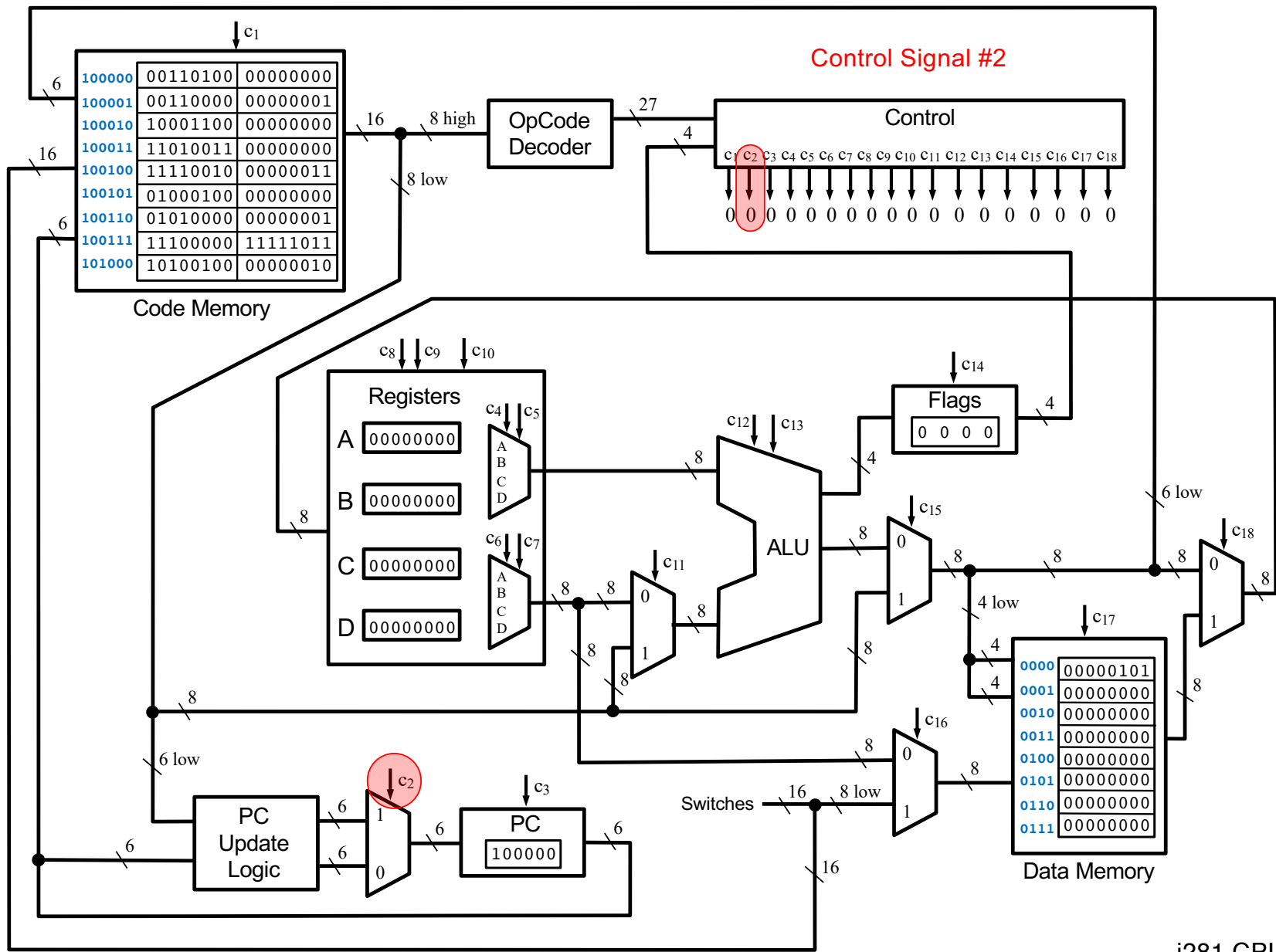
i281 CPU



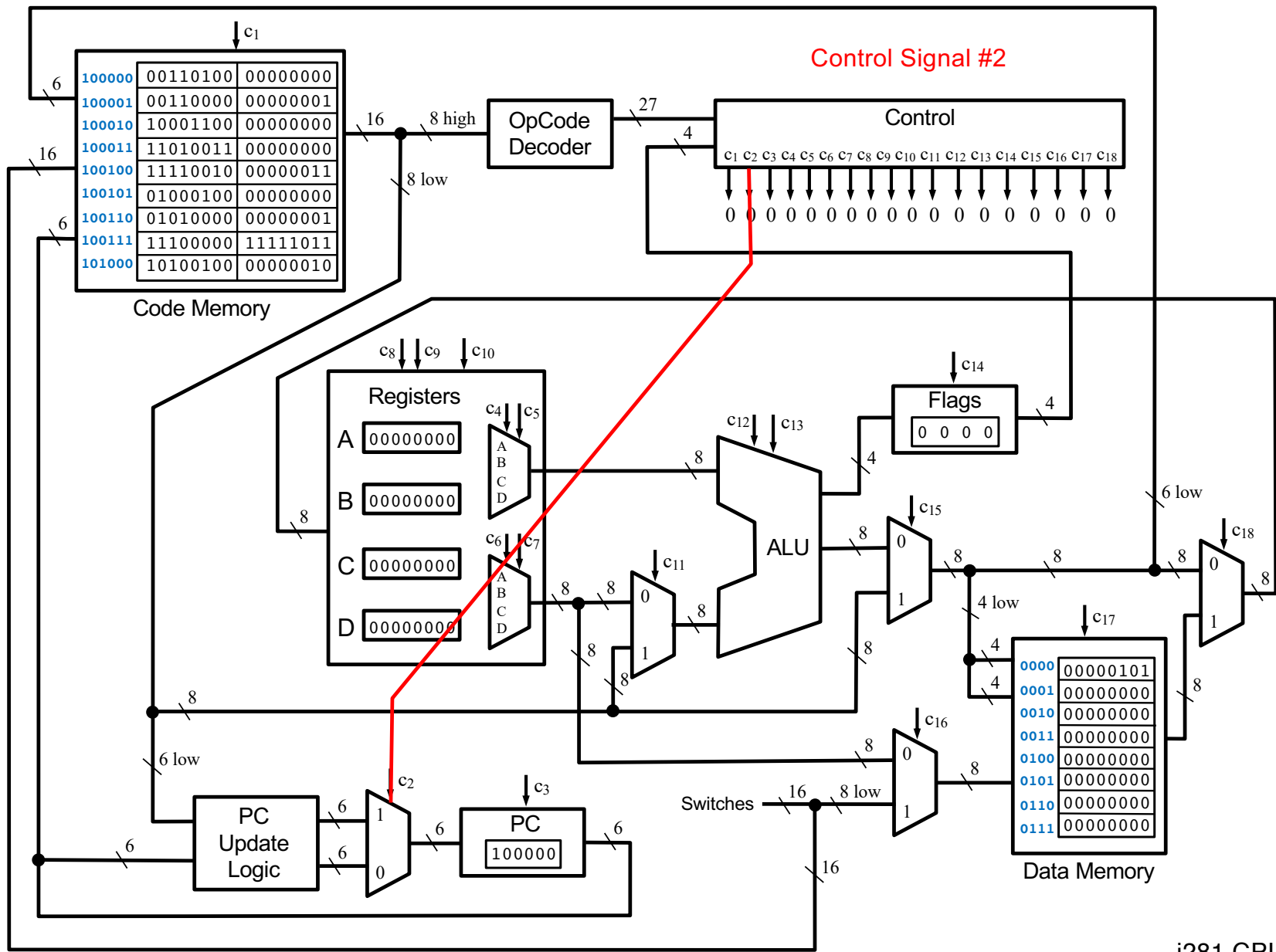
i281 CPU



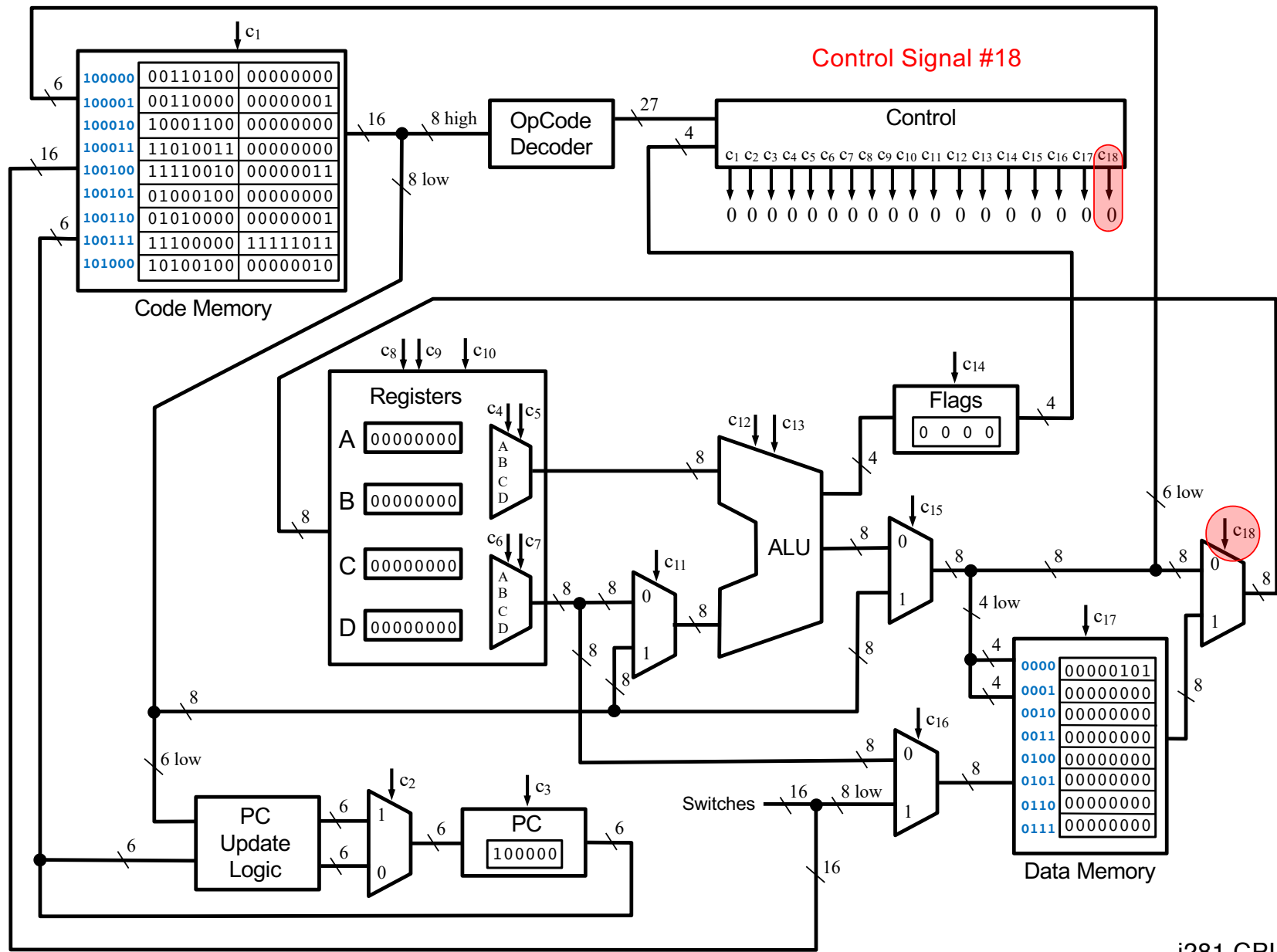
i281 CPU



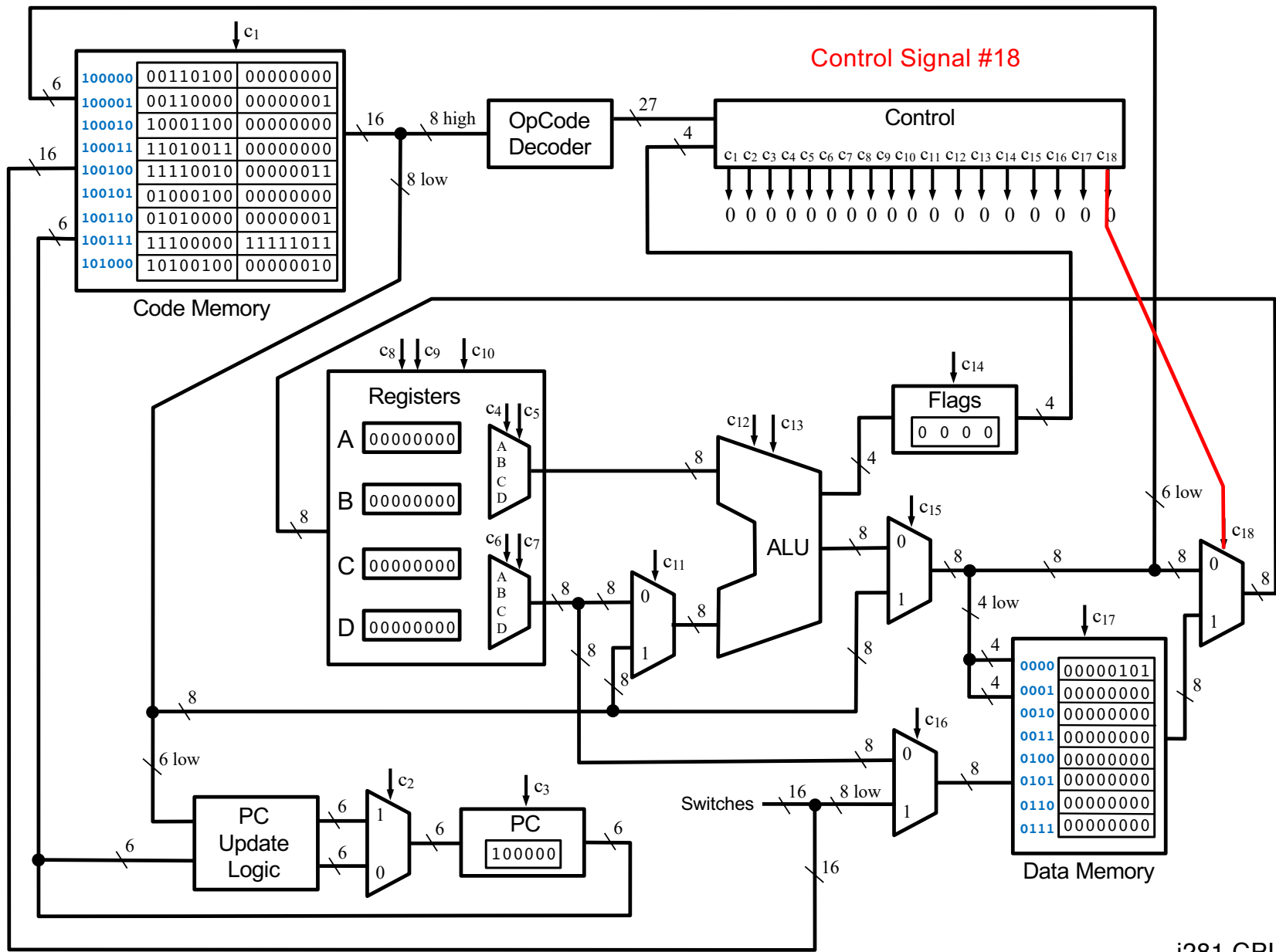
i281 CPU



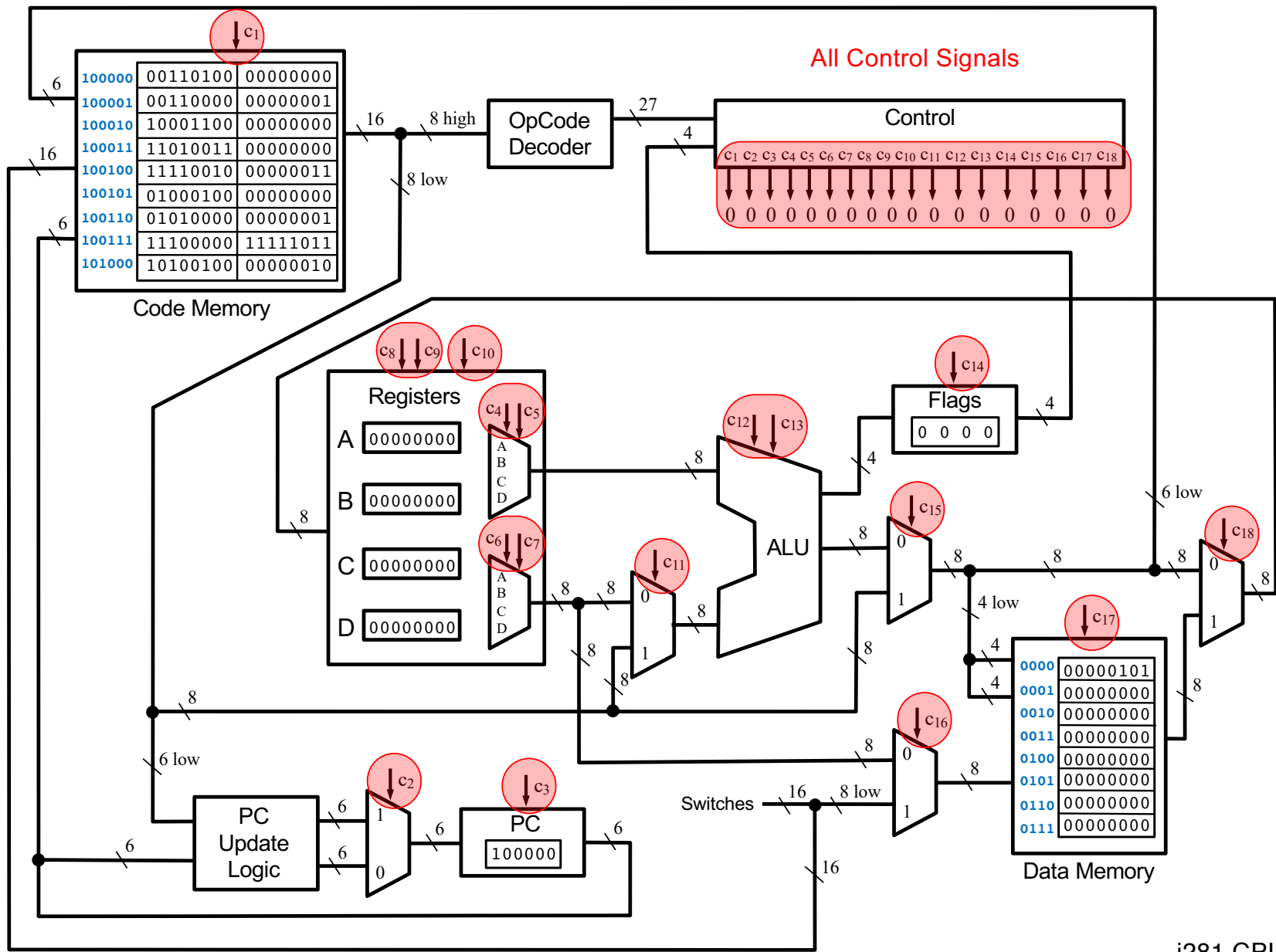
i281 CPU

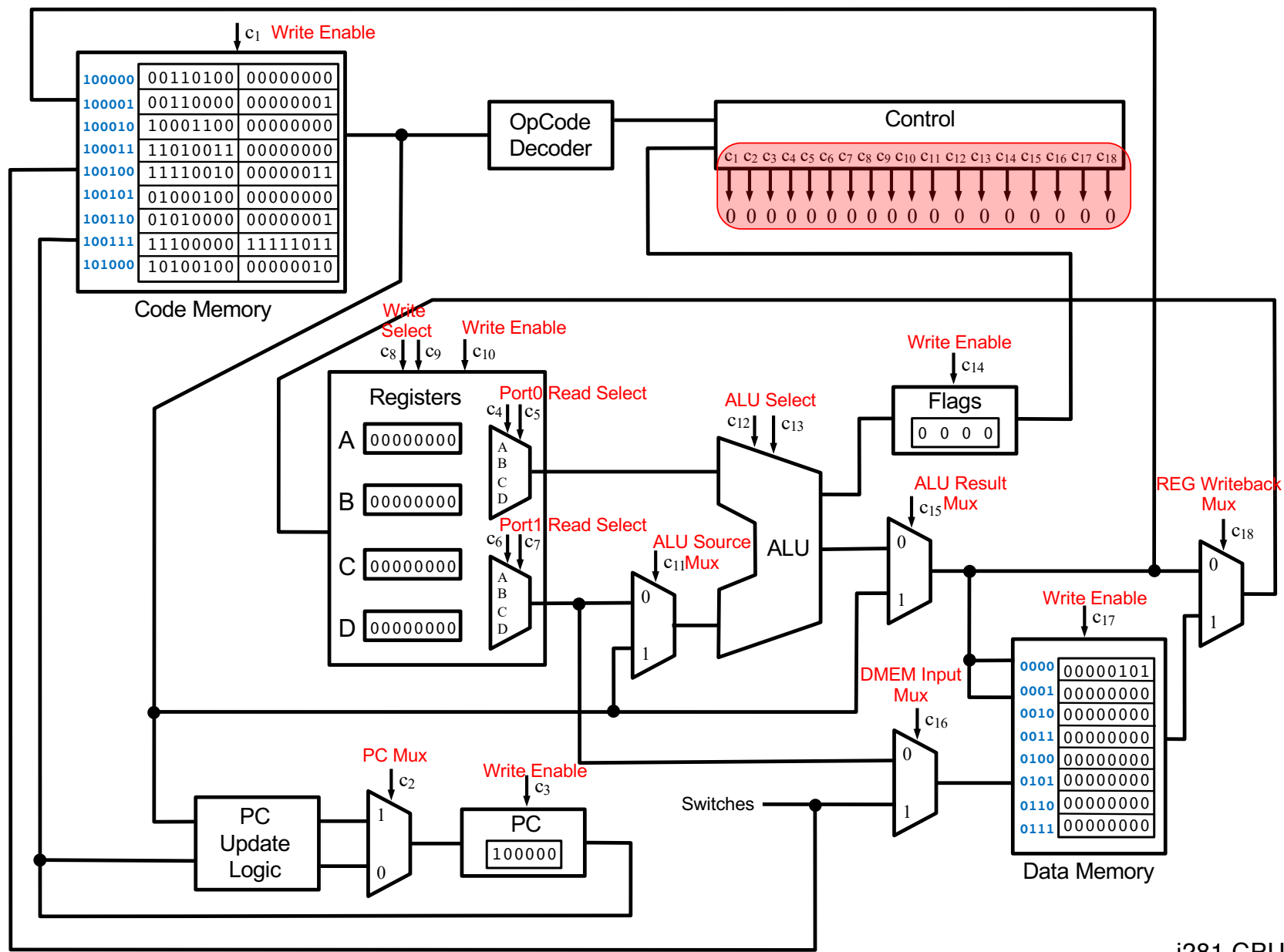


i281 CPU

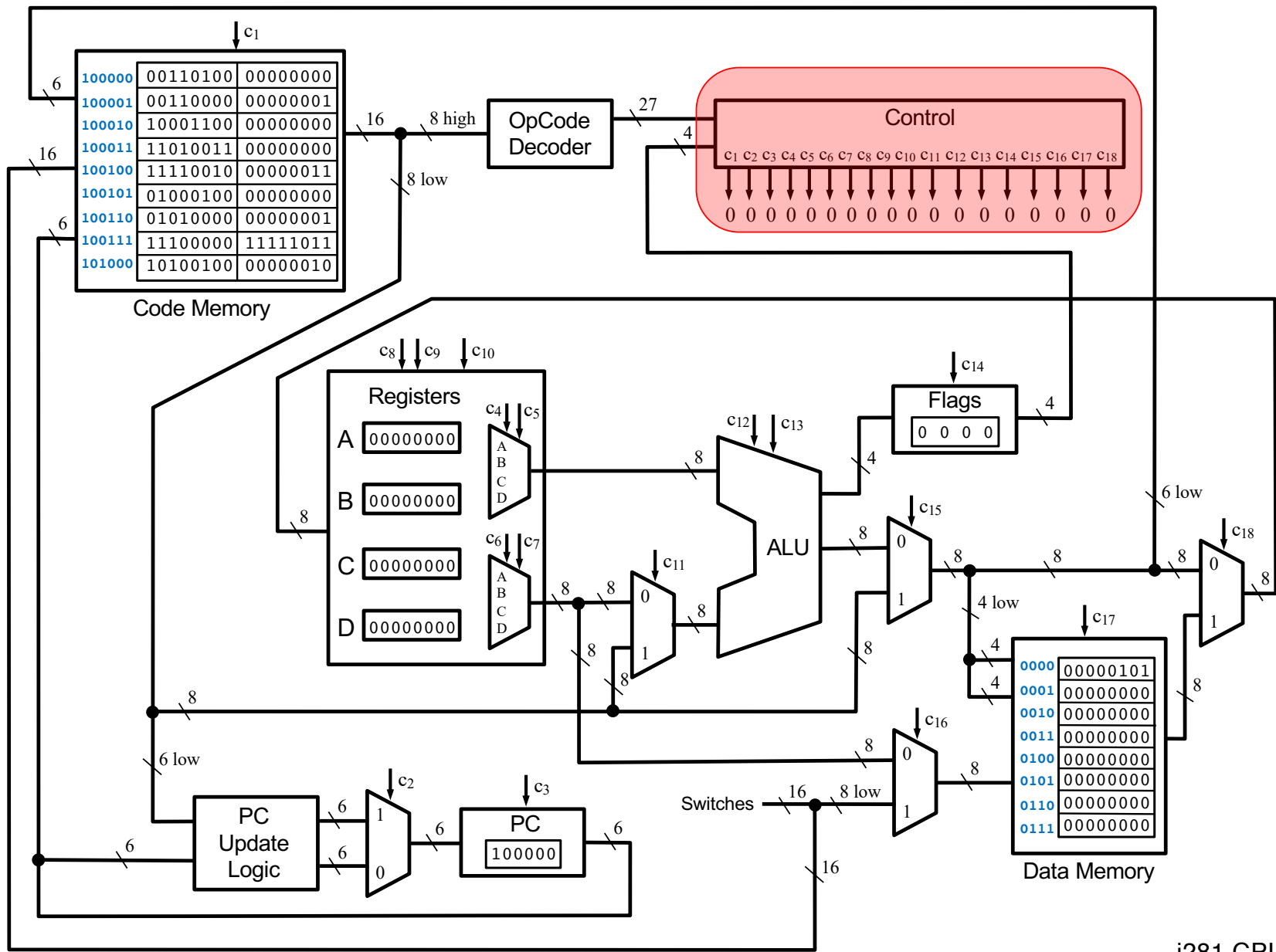


i281 CPU





i281 CPU



i281 CPU

The OPCODEs for this CPU

NOOP	NO OPeration
INPUTC	INPUT into Code memory
INPUTCF	INPUT into Code memory with oFfset
INPUTD	INPUT into Data memory
INPUTDF	INPUT into Data memory with oFfset
MOVE	MOVE the contents of one register into another
LOADI	LOAD Immediate value
LOADP	LOAD Pointer address
ADD	ADD two registers
ADDI	ADD an Immediate value to a register
SUB	SUBtract two registers
SUBI	SUBtract an Immediate value from a register
LOAD	LOAD from a data memory address into a register
LOADF	LOAD with an oFfset specified by another register
STORE	STORE a register into a data memory address
STOREF	STORE with an oFfset specified by another register
SHIFTL	SHIFT Left all bits in a register
SHIFTR	SHIFT Right all bits in a register
CMP	CoMPare the values in two registers
JUMP	JUMP unconditionally to a specified address
BRE	BRanch if Equal
BRZ	BRanch if Zero
BRNE	BRanch if Not Equal
BRNZ	BRanch if Not Zero
BRG	BRanch if Greater
BRGE	BRanch if Greater than or Equal

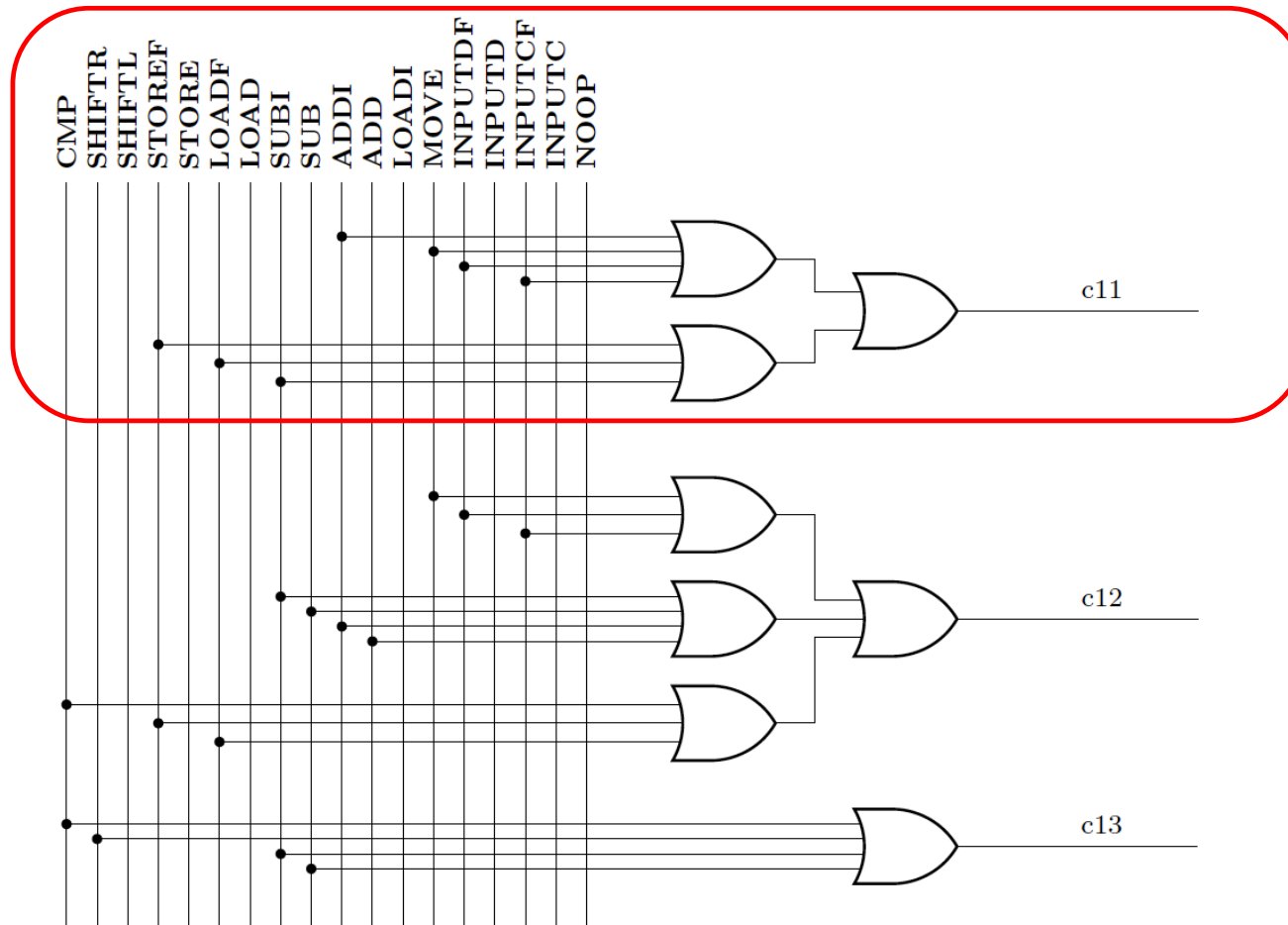
[illegible]

18 control lines

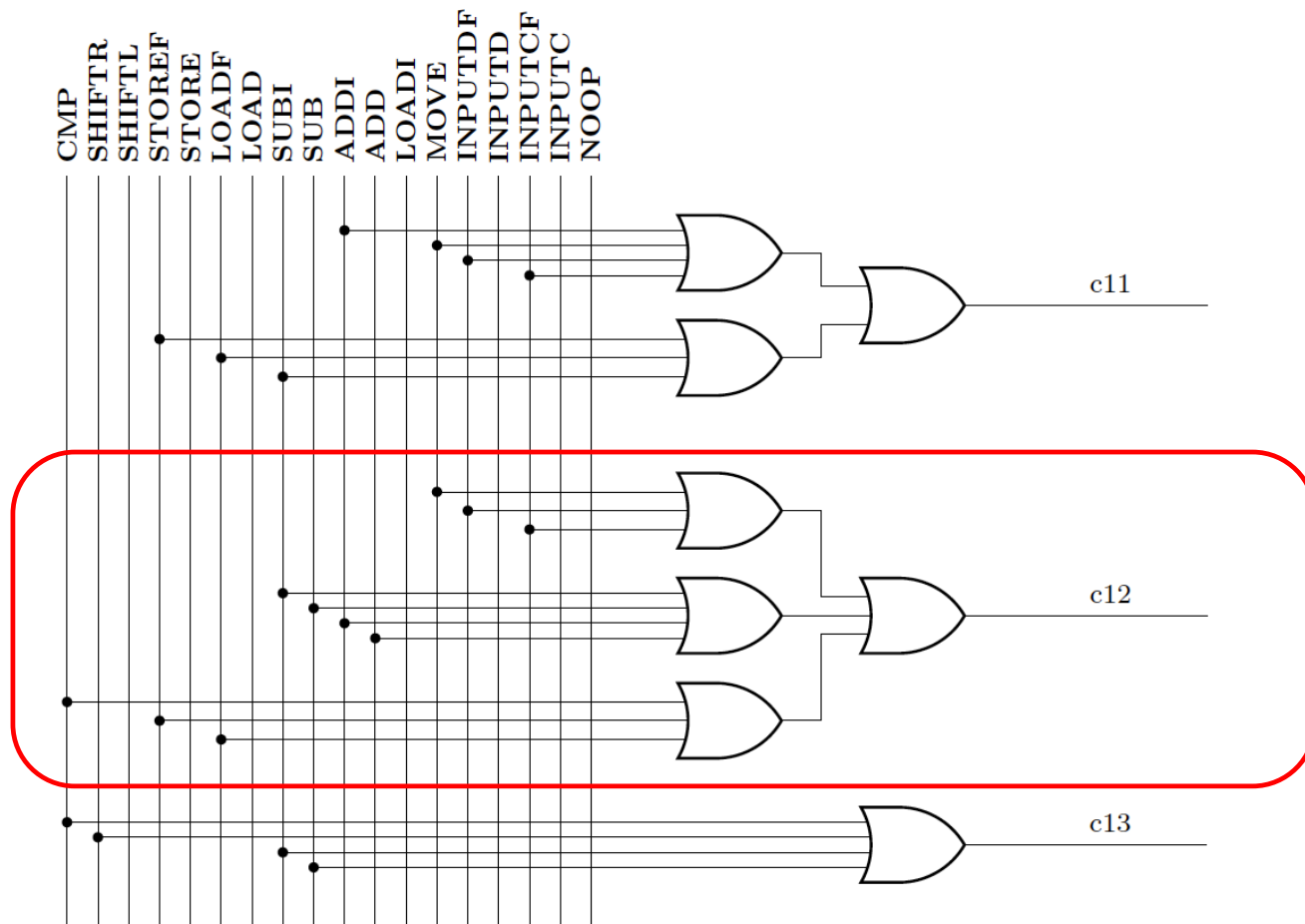
23 one-hot
encoded
OPCODEs

[illegible]

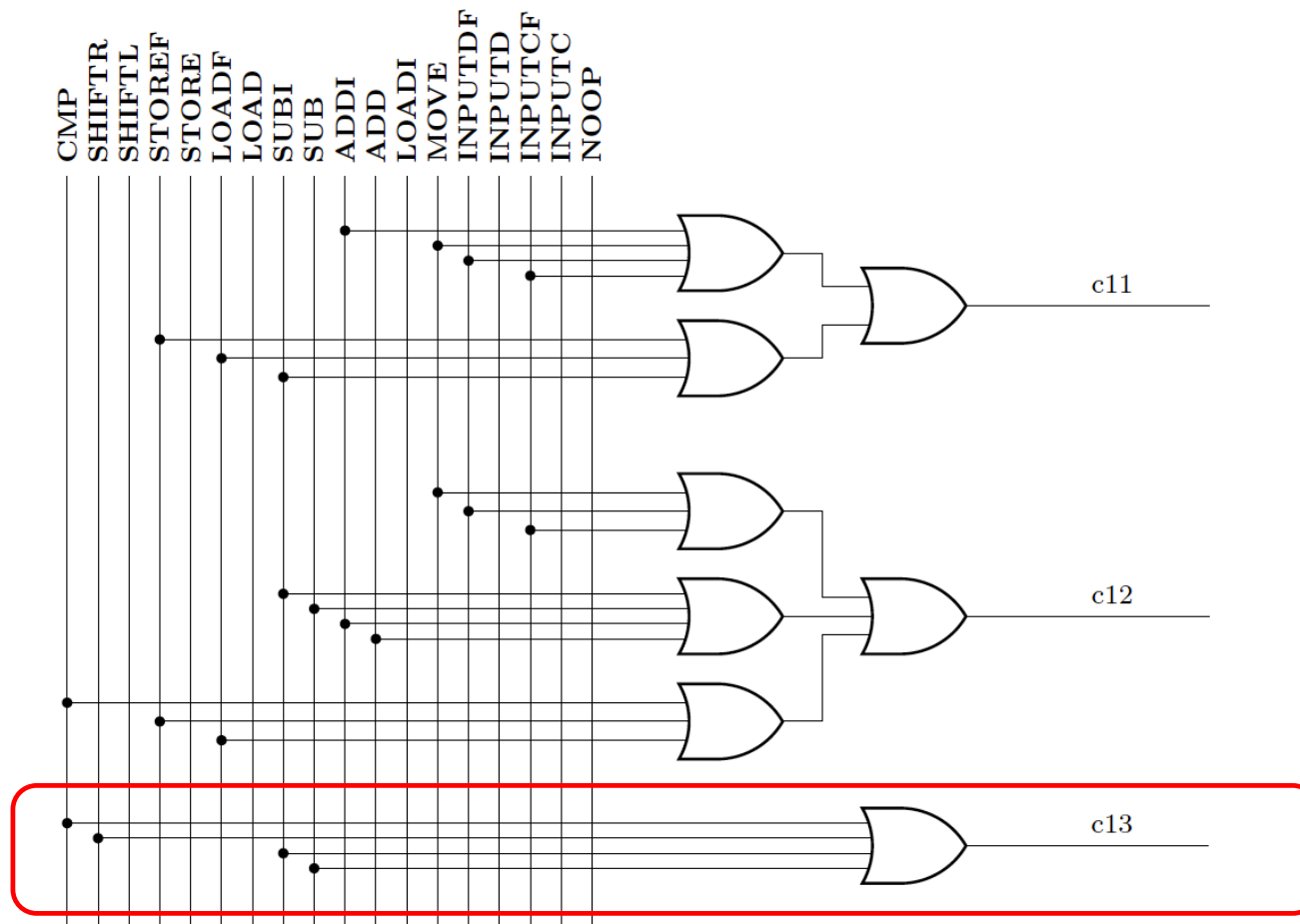
The Wiring Diagram for c_{11}



The Wiring Diagram for c_{12}



The Wiring Diagram for c_{13}



All others are set to zero.

[illegible]

C₈ and C₉ depend
on the instruction and
the register that it uses.

[illegible]

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:

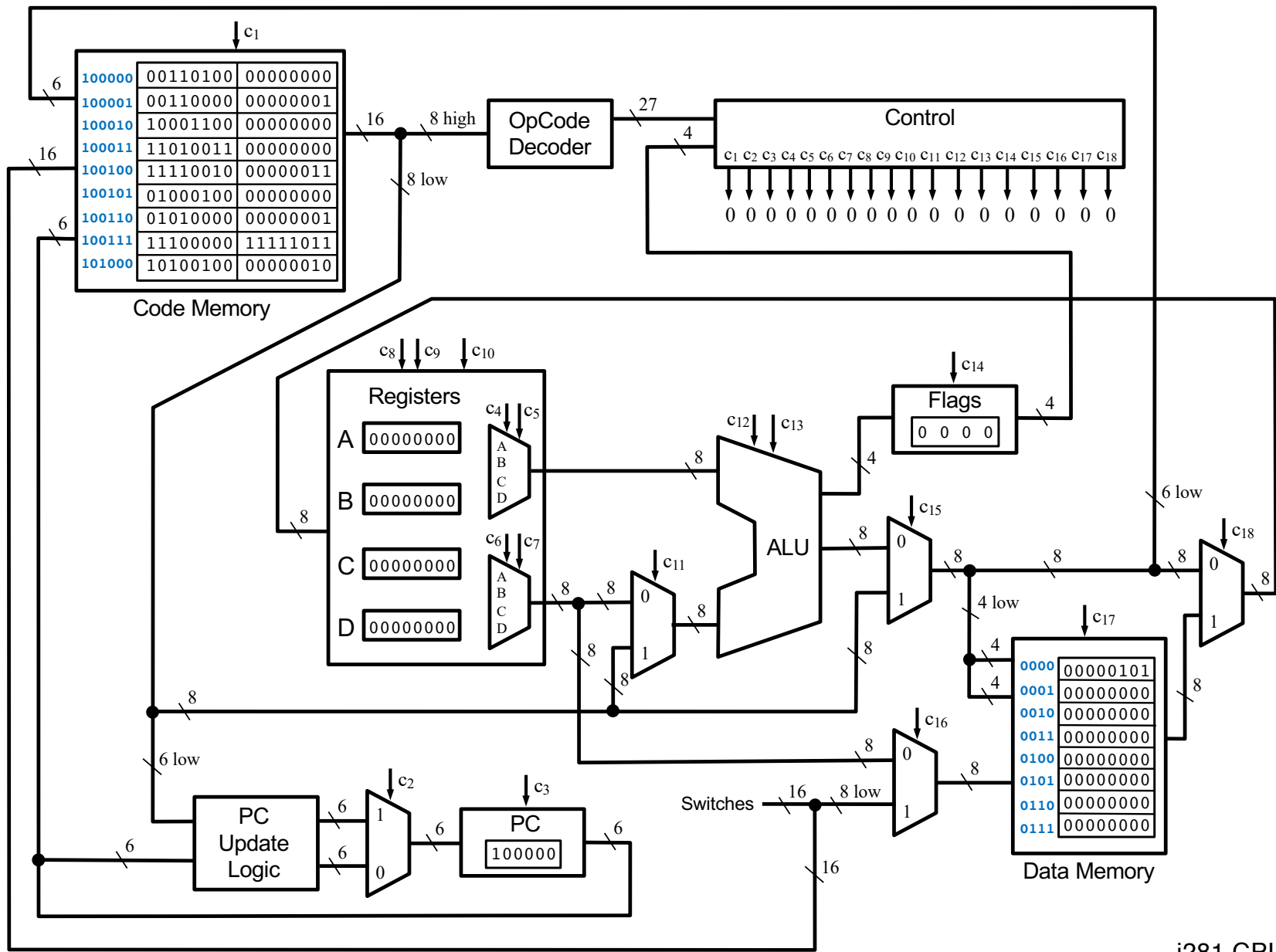
00000101
00000000
00000000

.code

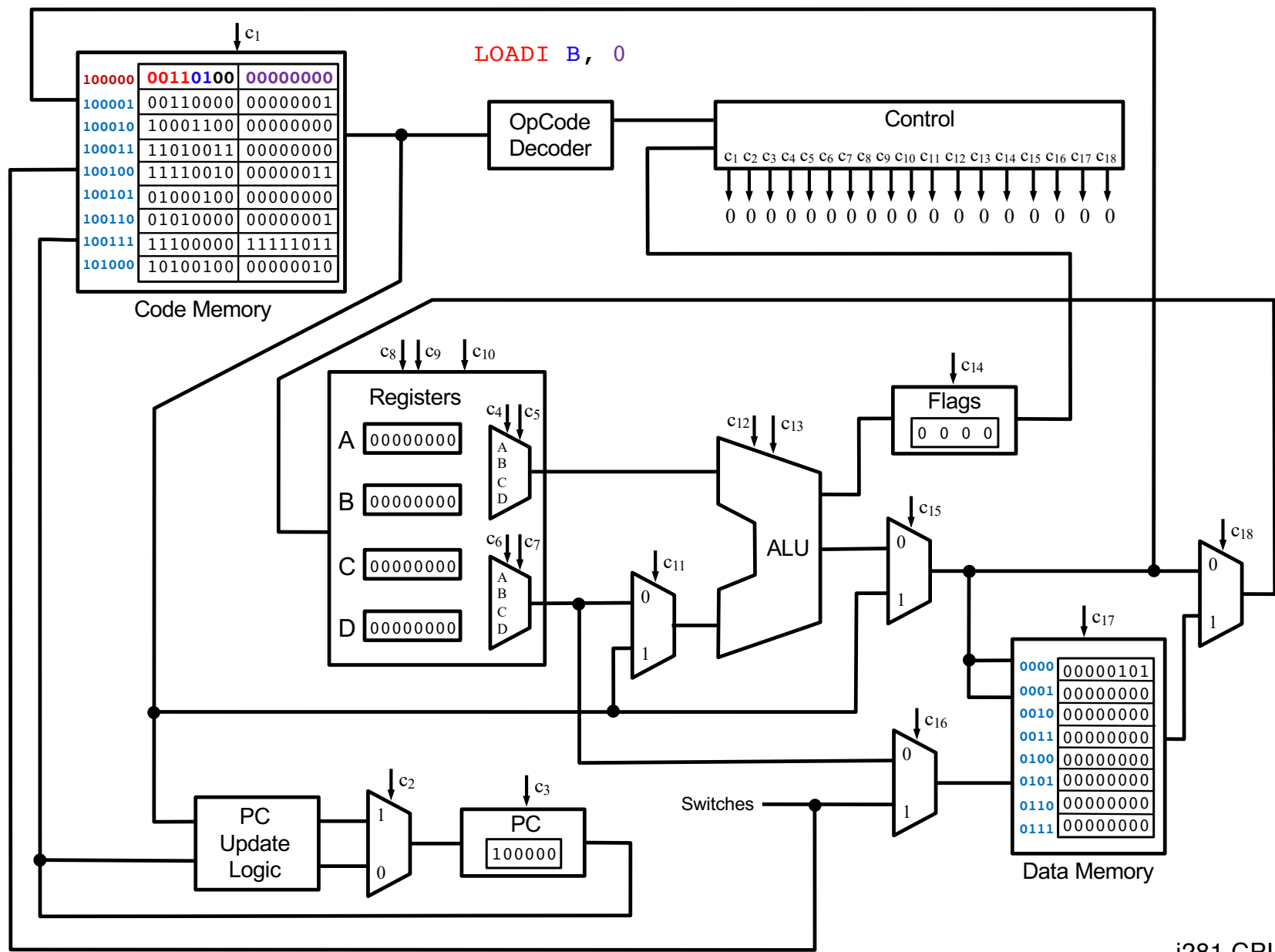
LOADI **B**, **0**
 LOADI **A**, **1**
 LOAD **D**, [**N**]
Loop: **CMP** **A**, **D**
 BRG **End**
Add: **ADD** **B**, **A**
 ADDI **A**, **1**
 JUMP **Loop**
End: **STORE** [**sum**], **B**

Code Memory:

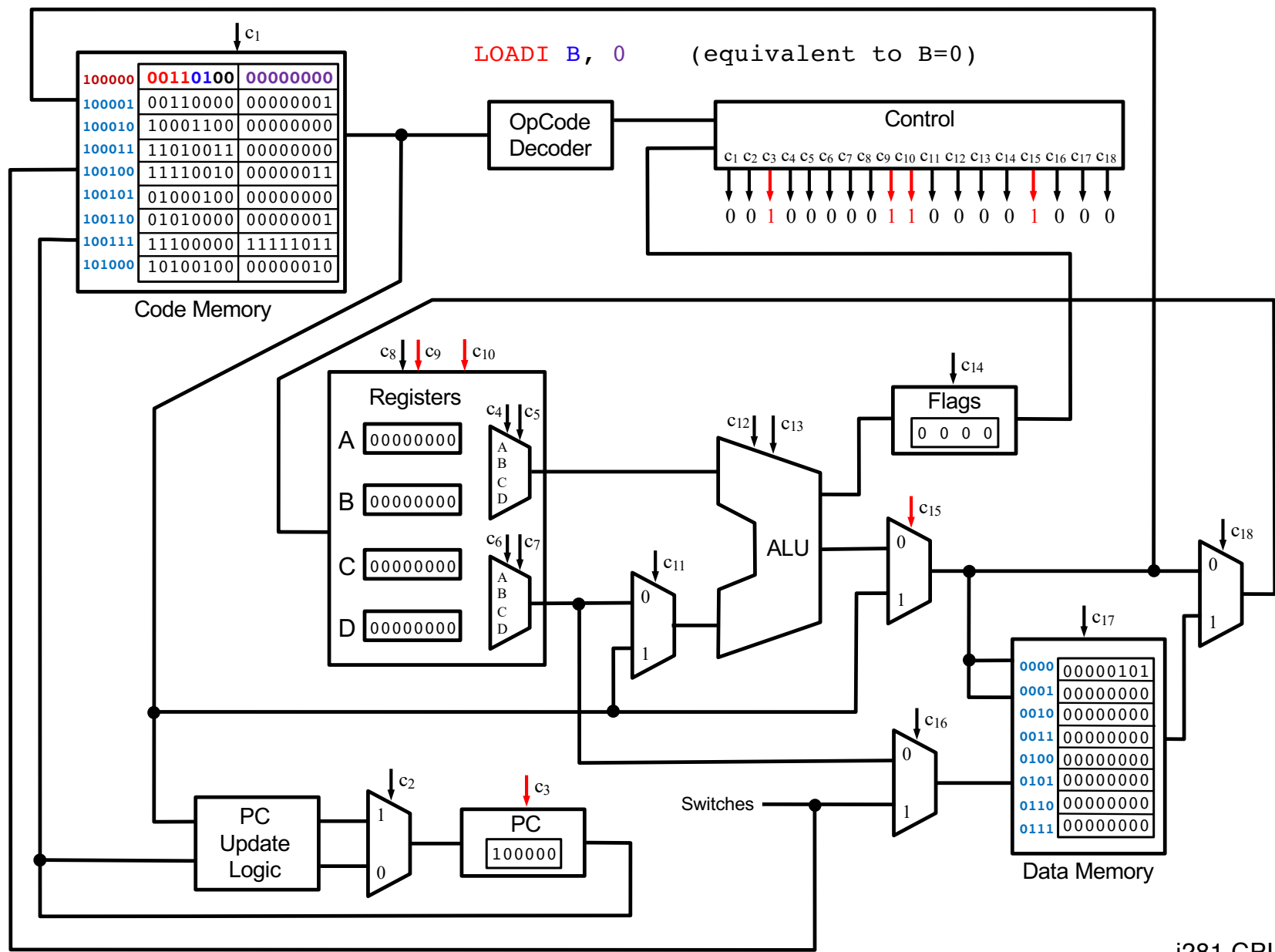
0011_01_00_00000000
0011_00_00_00000001
1000_11_00_00000000
1101_00_11_00000000
1111_00_10_00000011
0100_01_00_00000000
0101_00_00_00000001
1110_00_00_11111011
1010_01_00_00000010



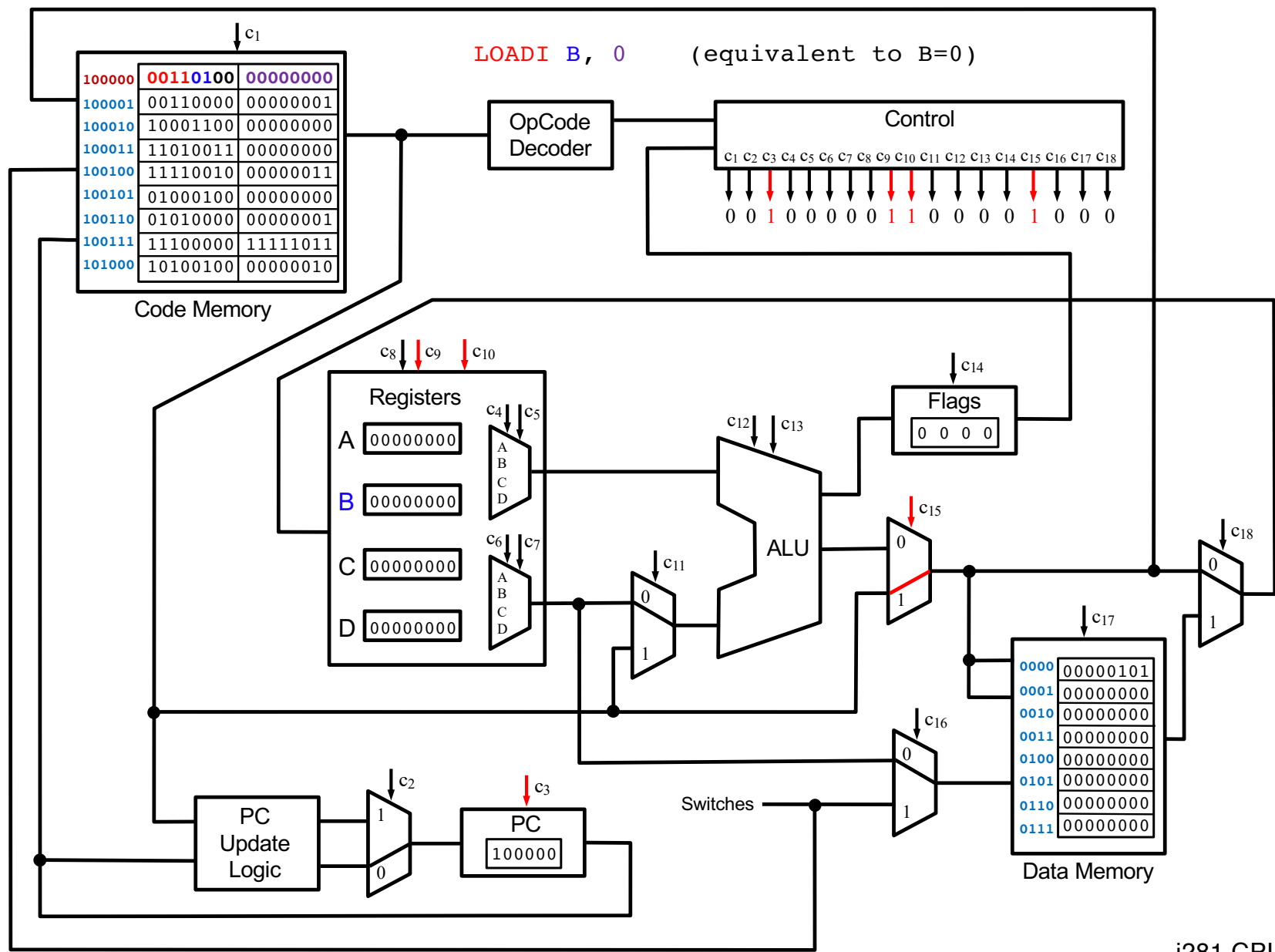
i281 CPU



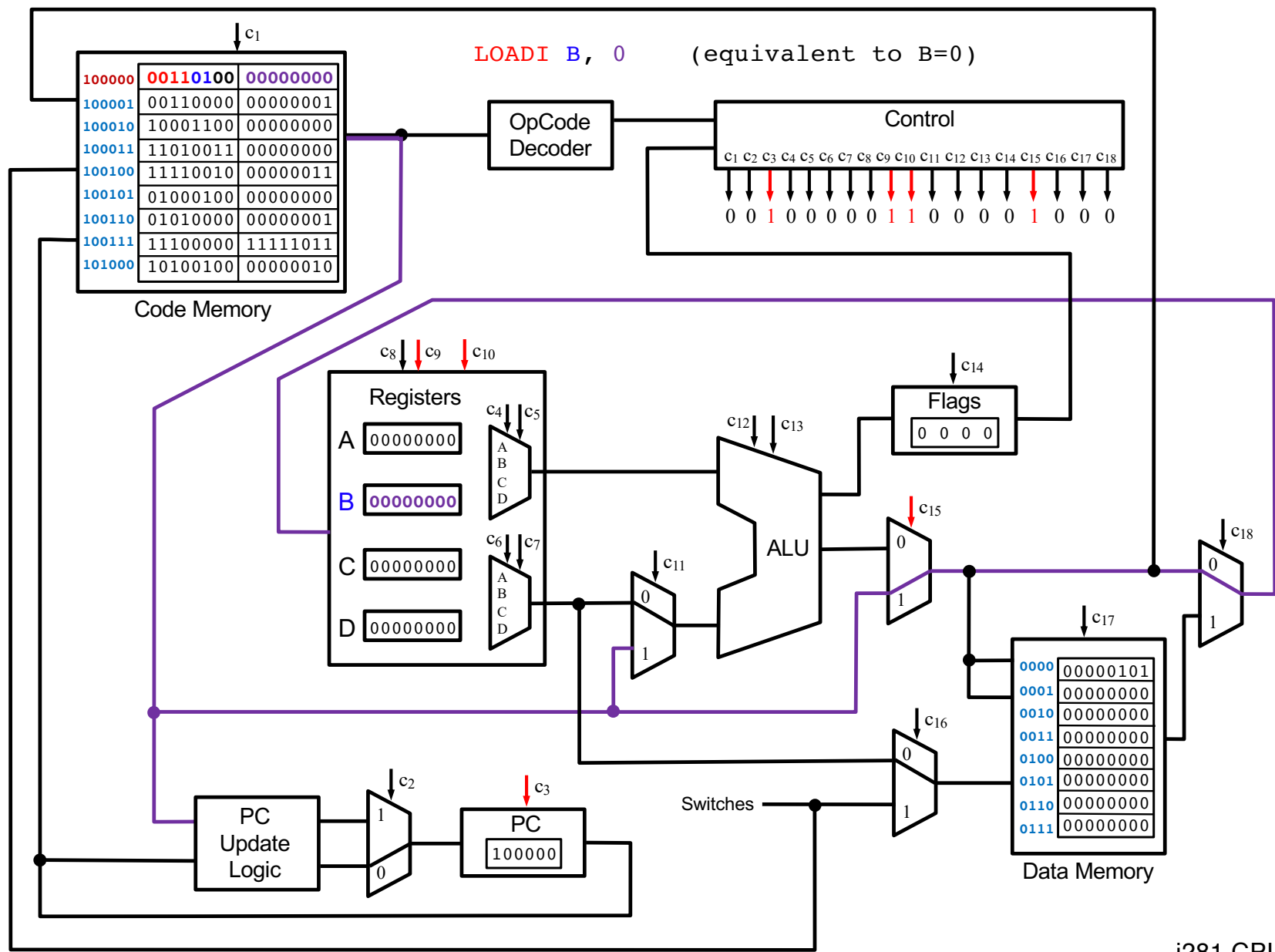
i281 CPU

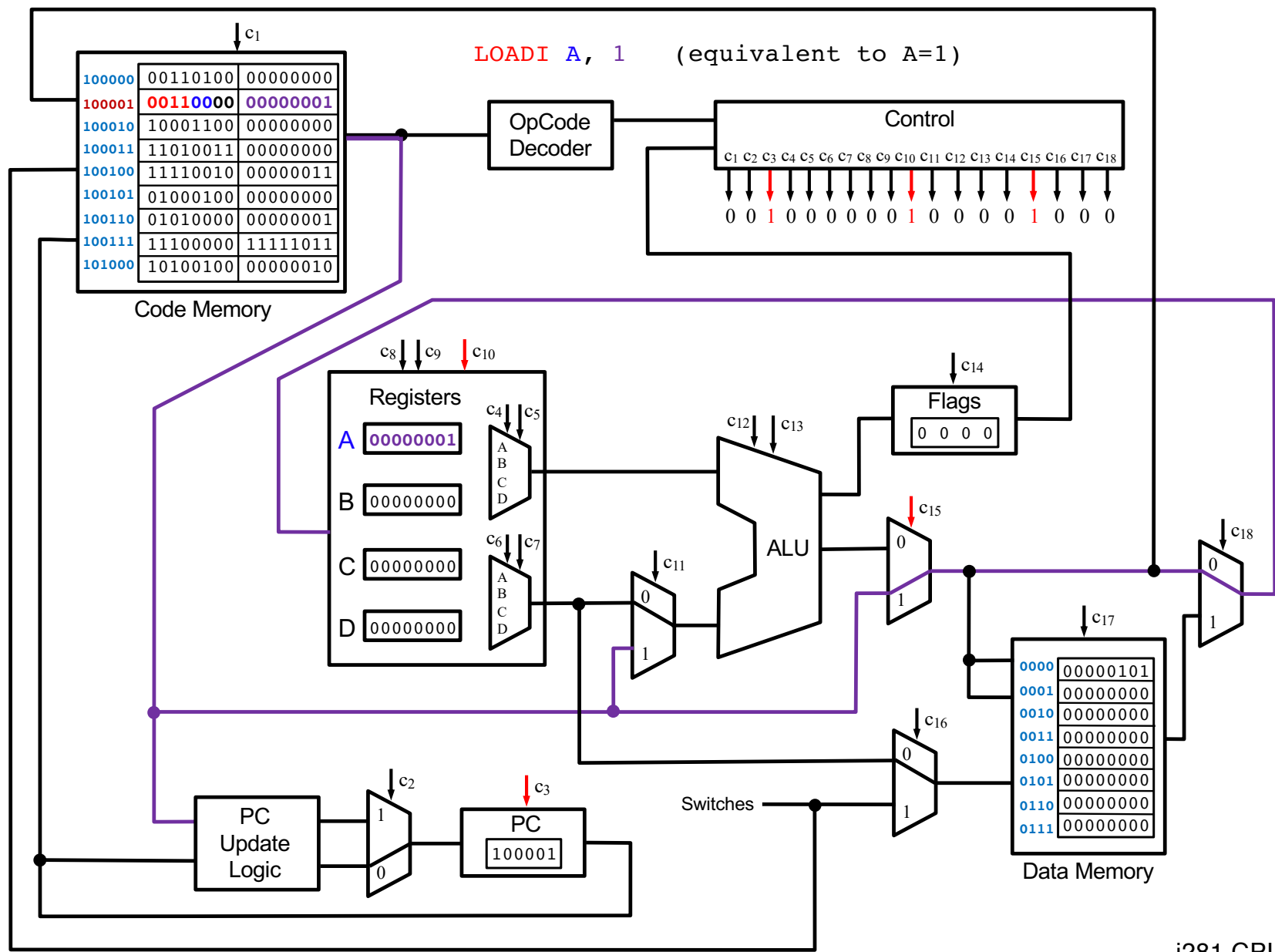


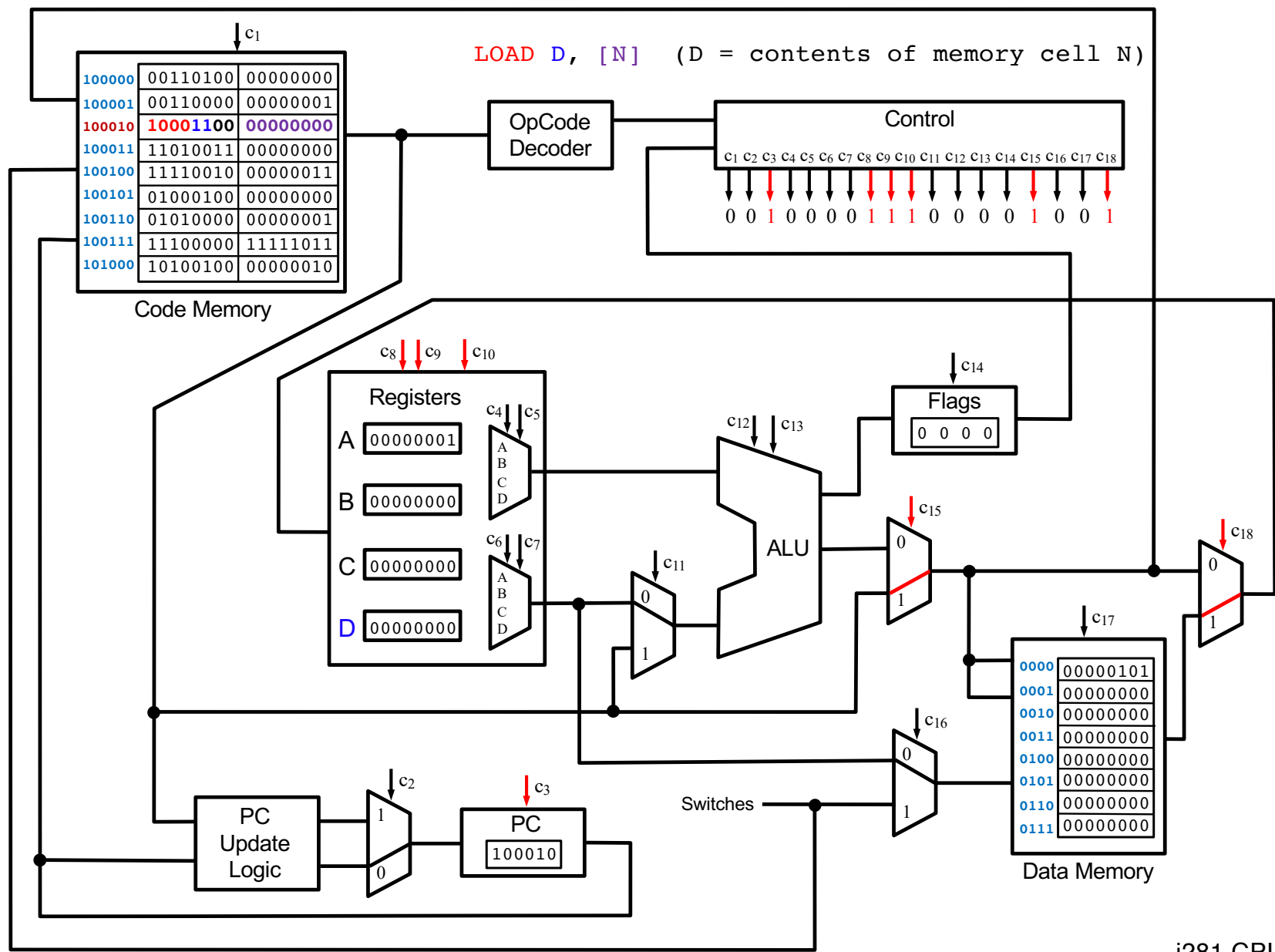
i281 CPU



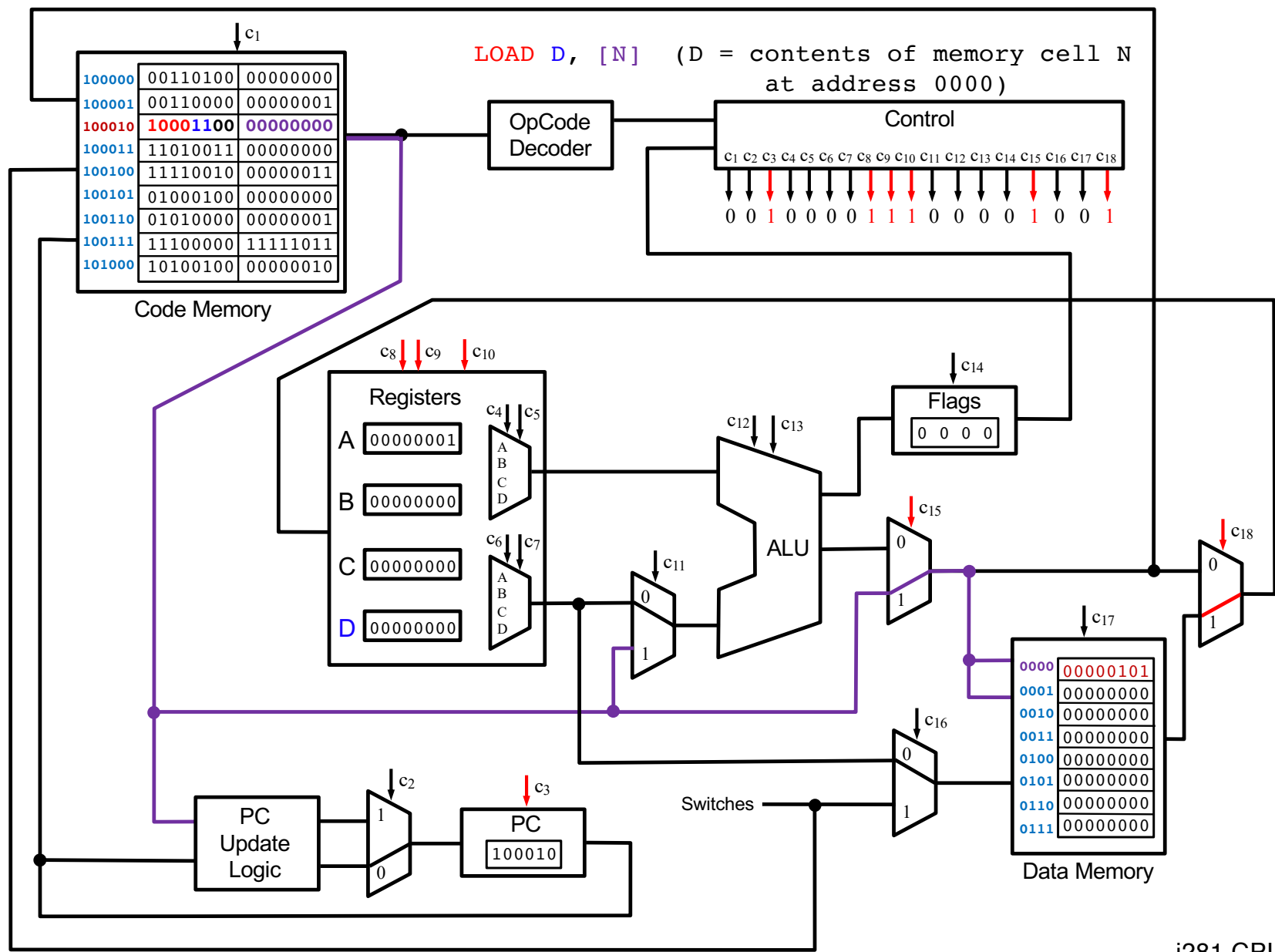
i281 CPU



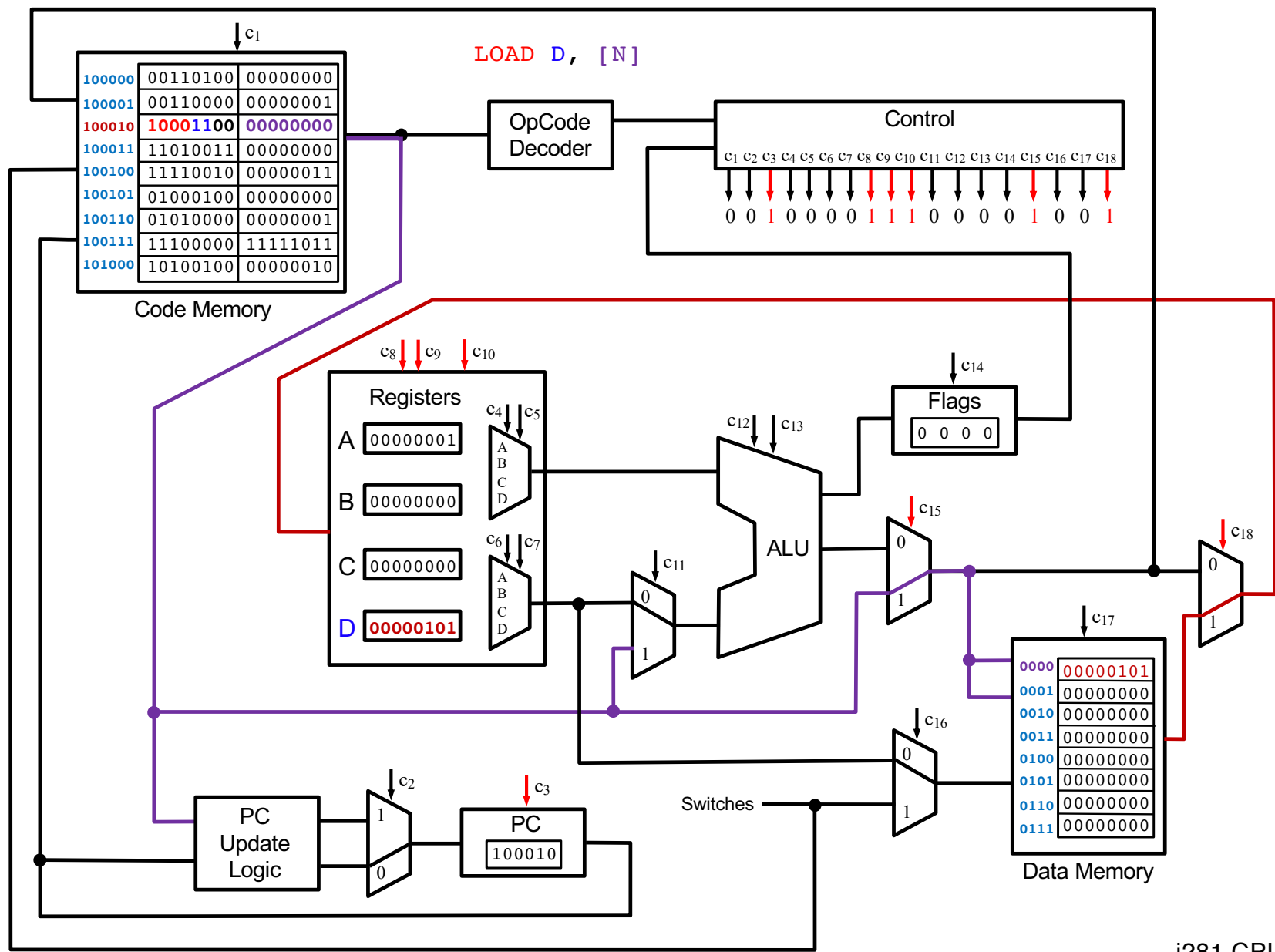




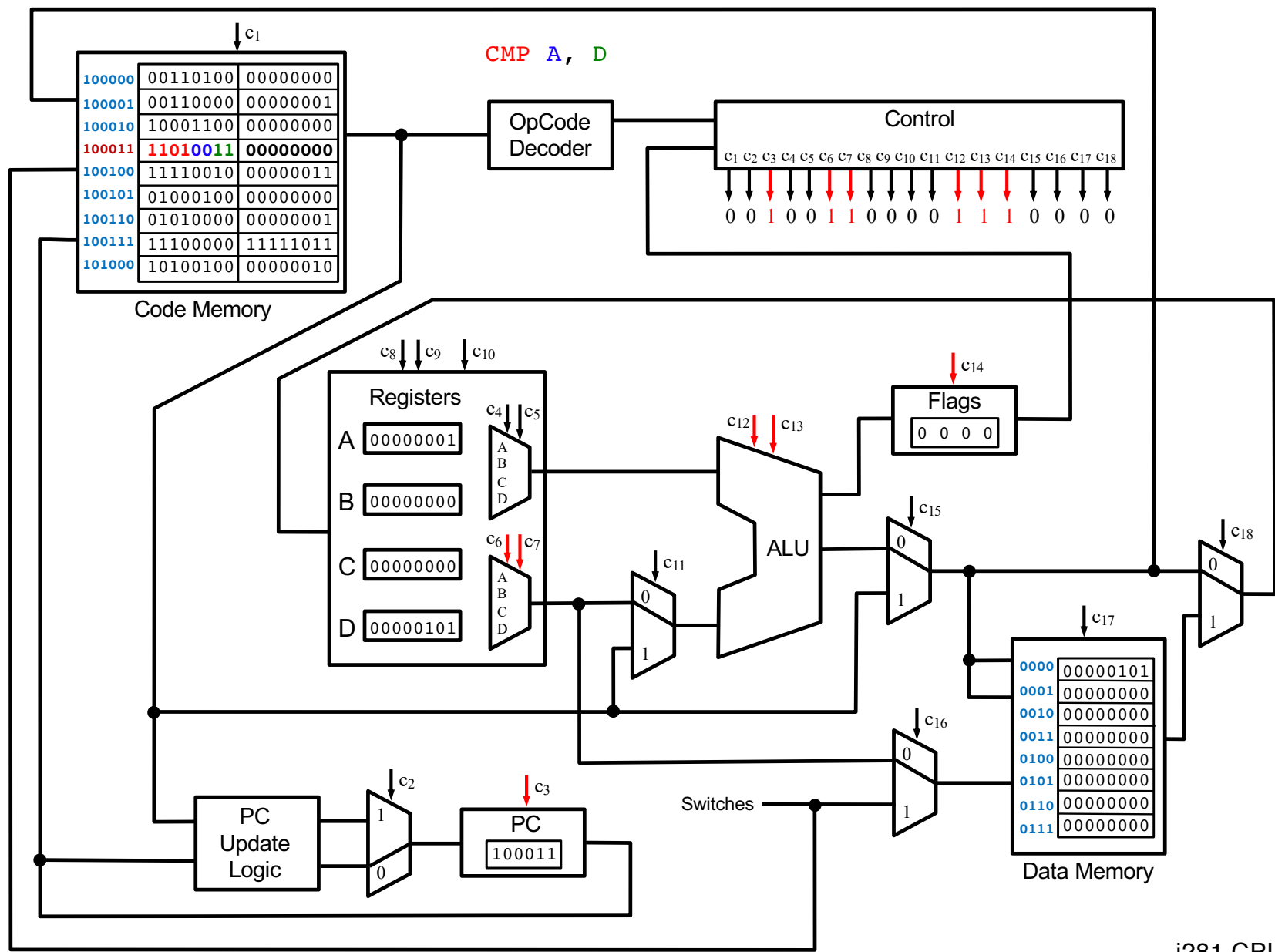
i281 CPU



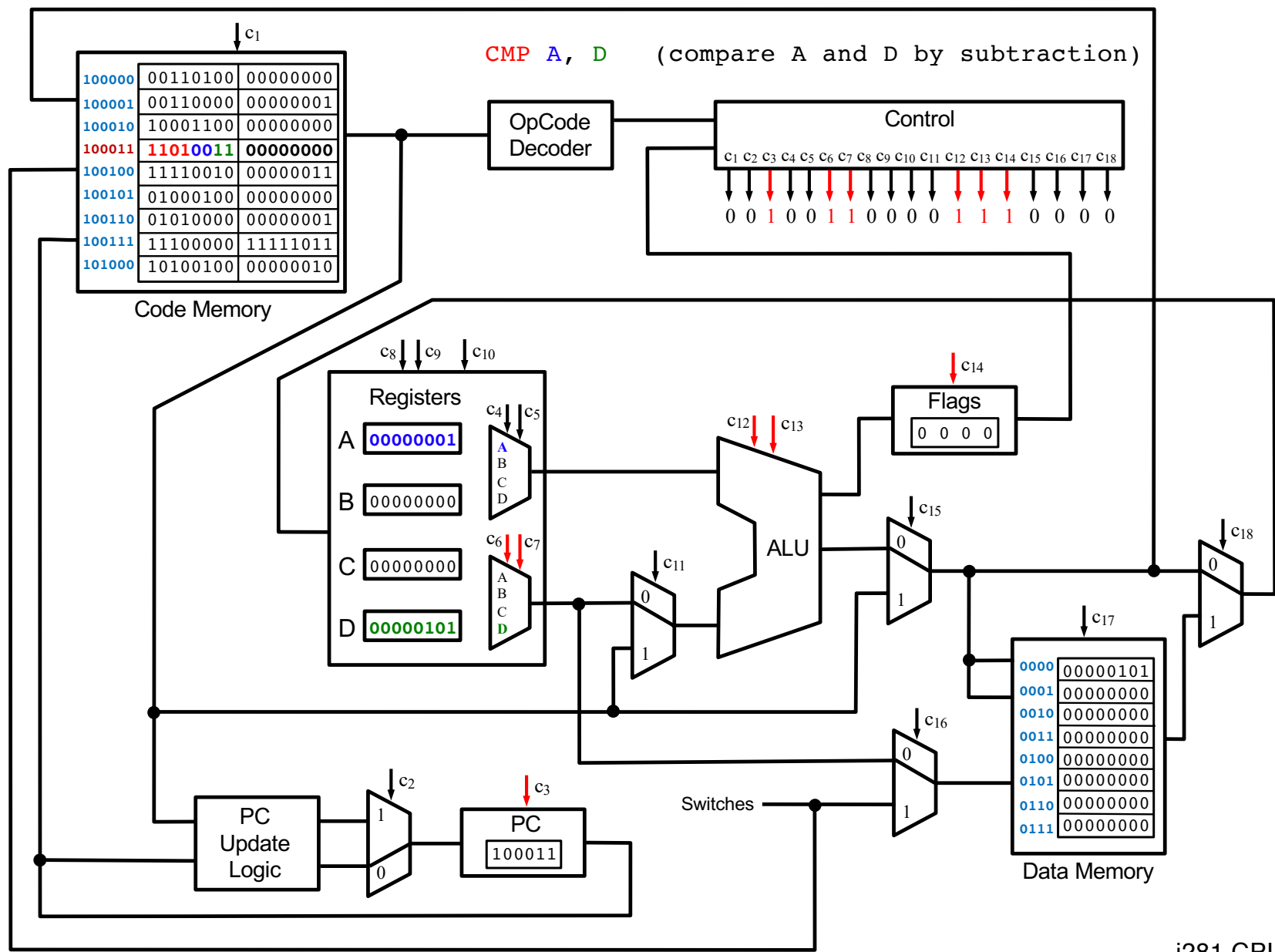
i281 CPU

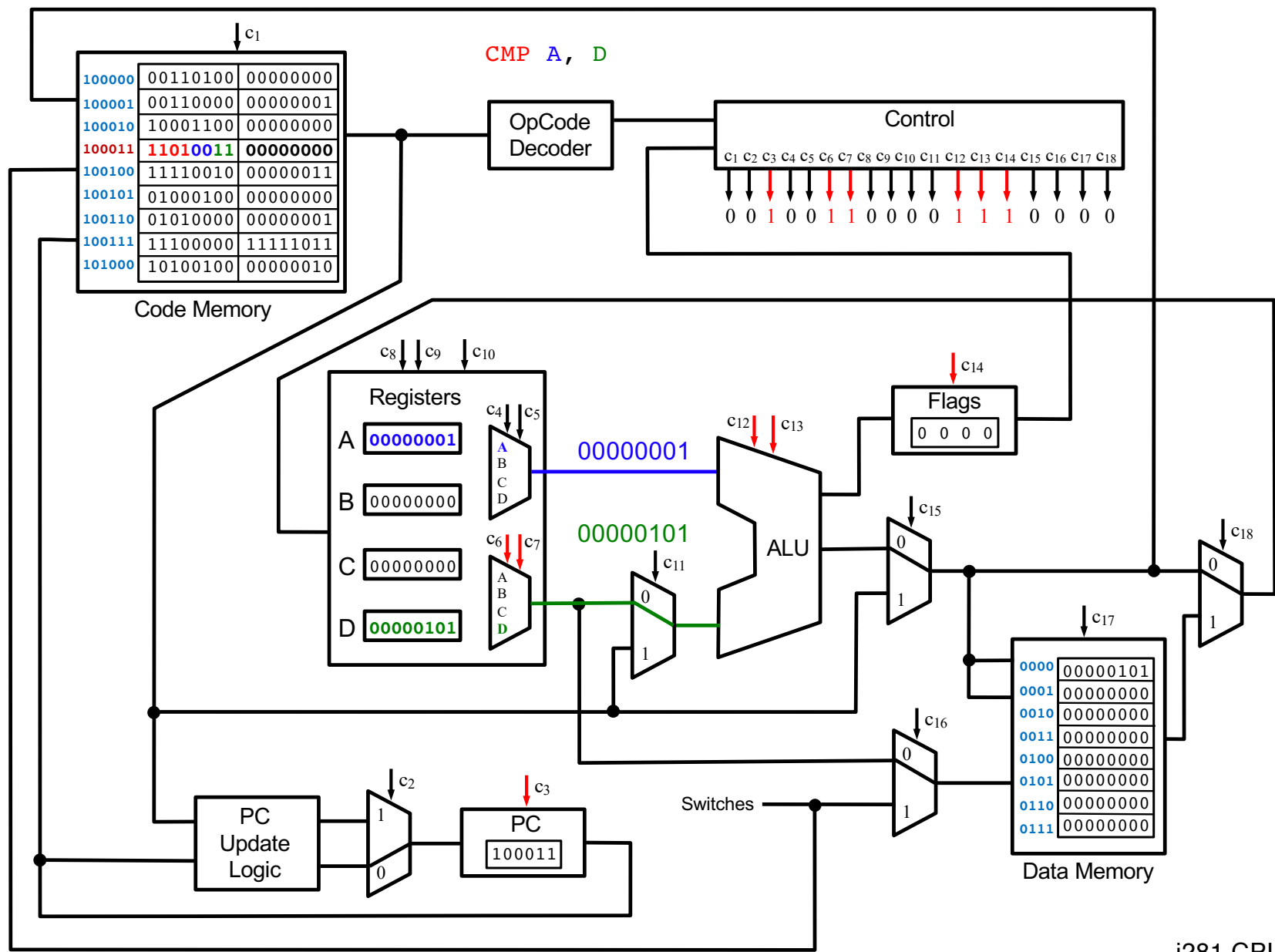


i281 CPU

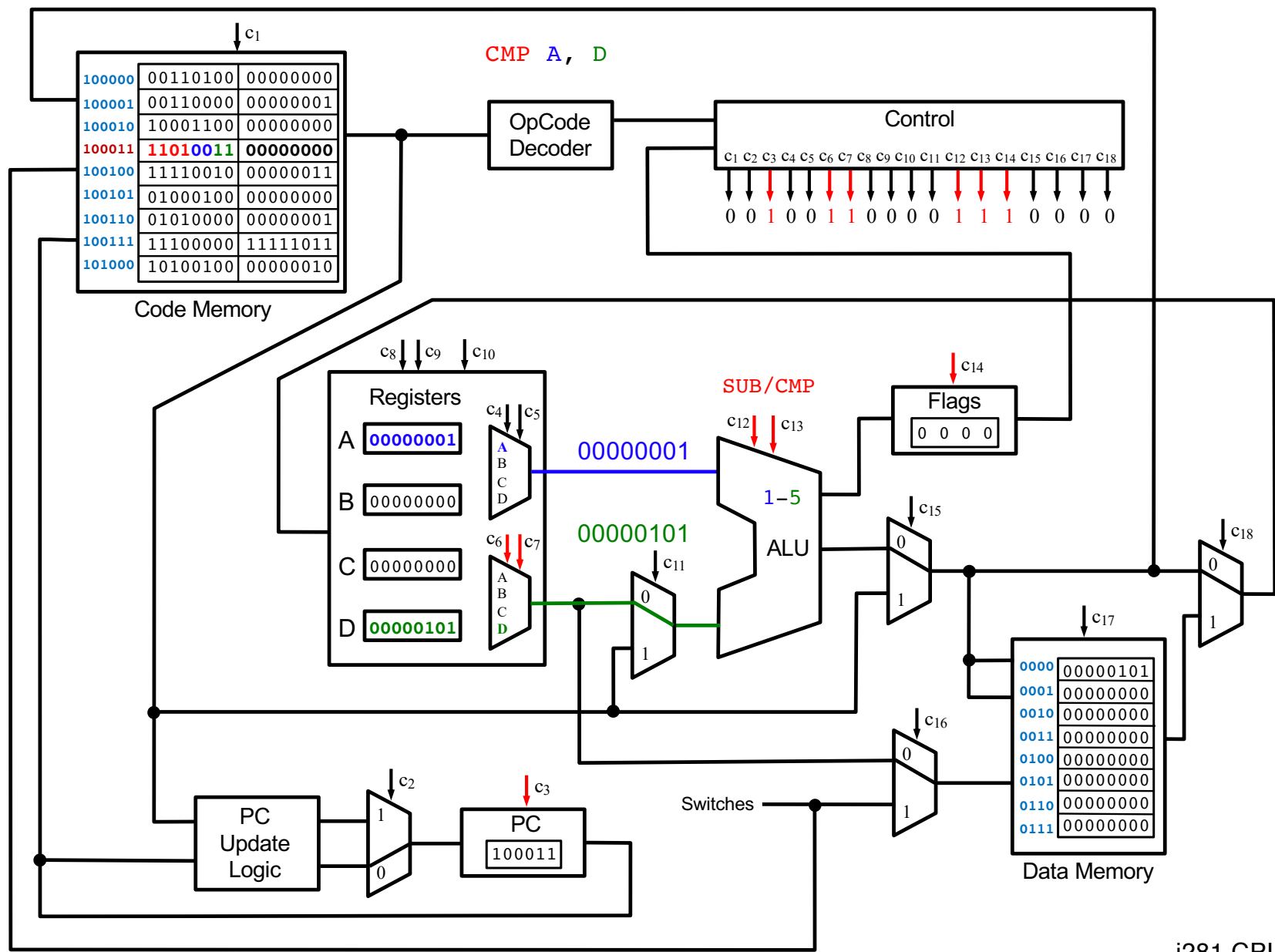


i281 CPU

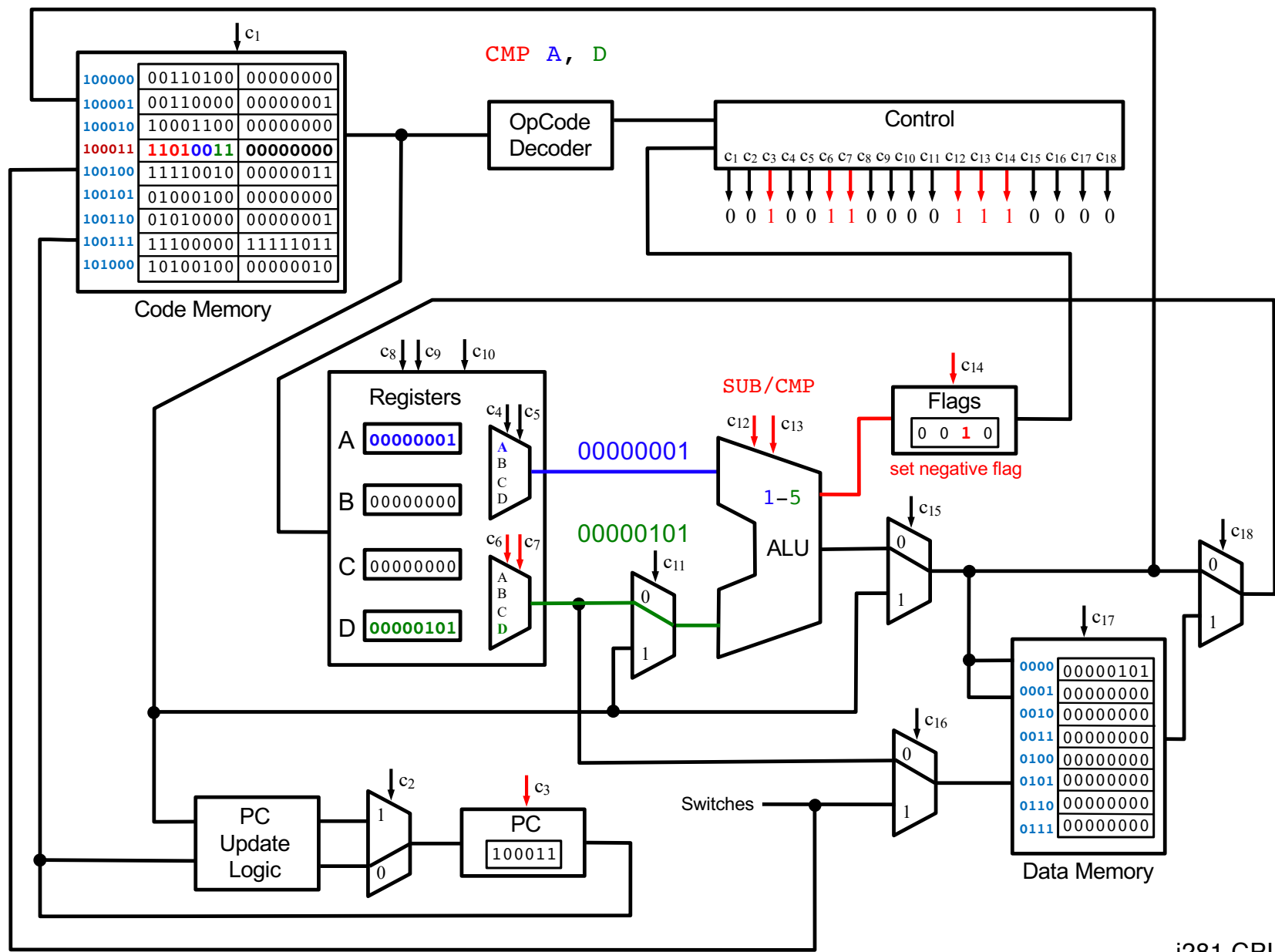




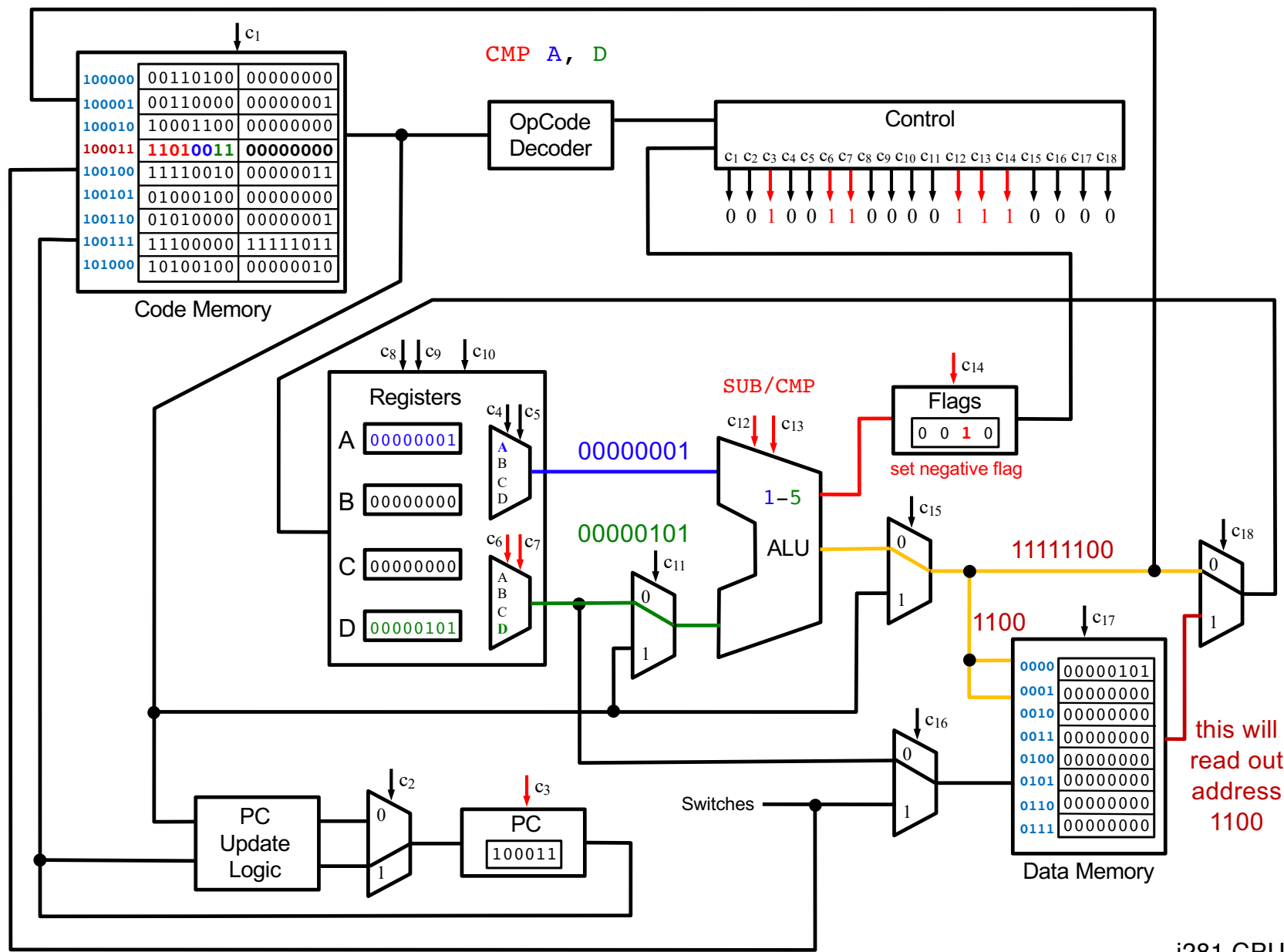
i281 CPU



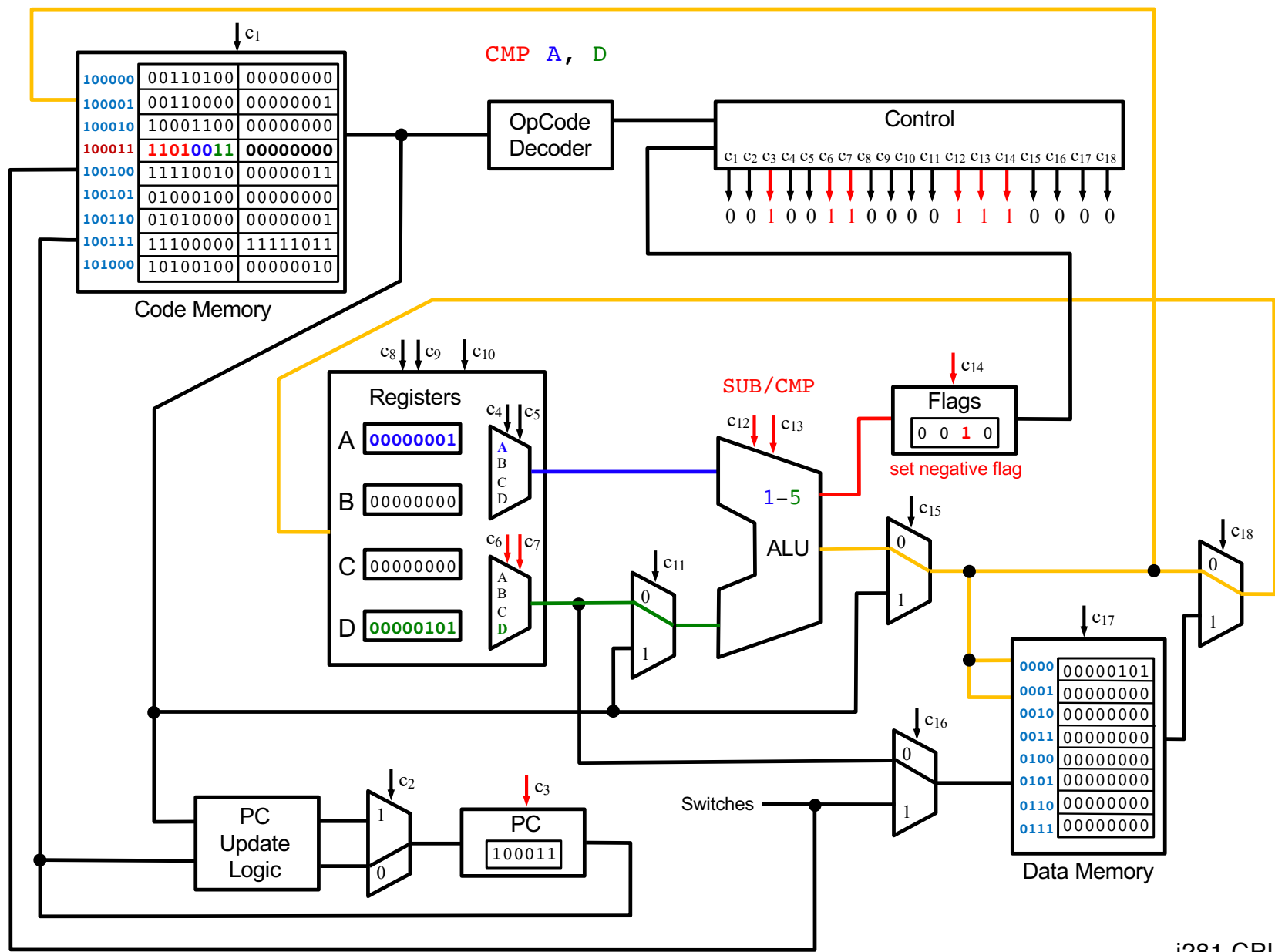
i281 CPU



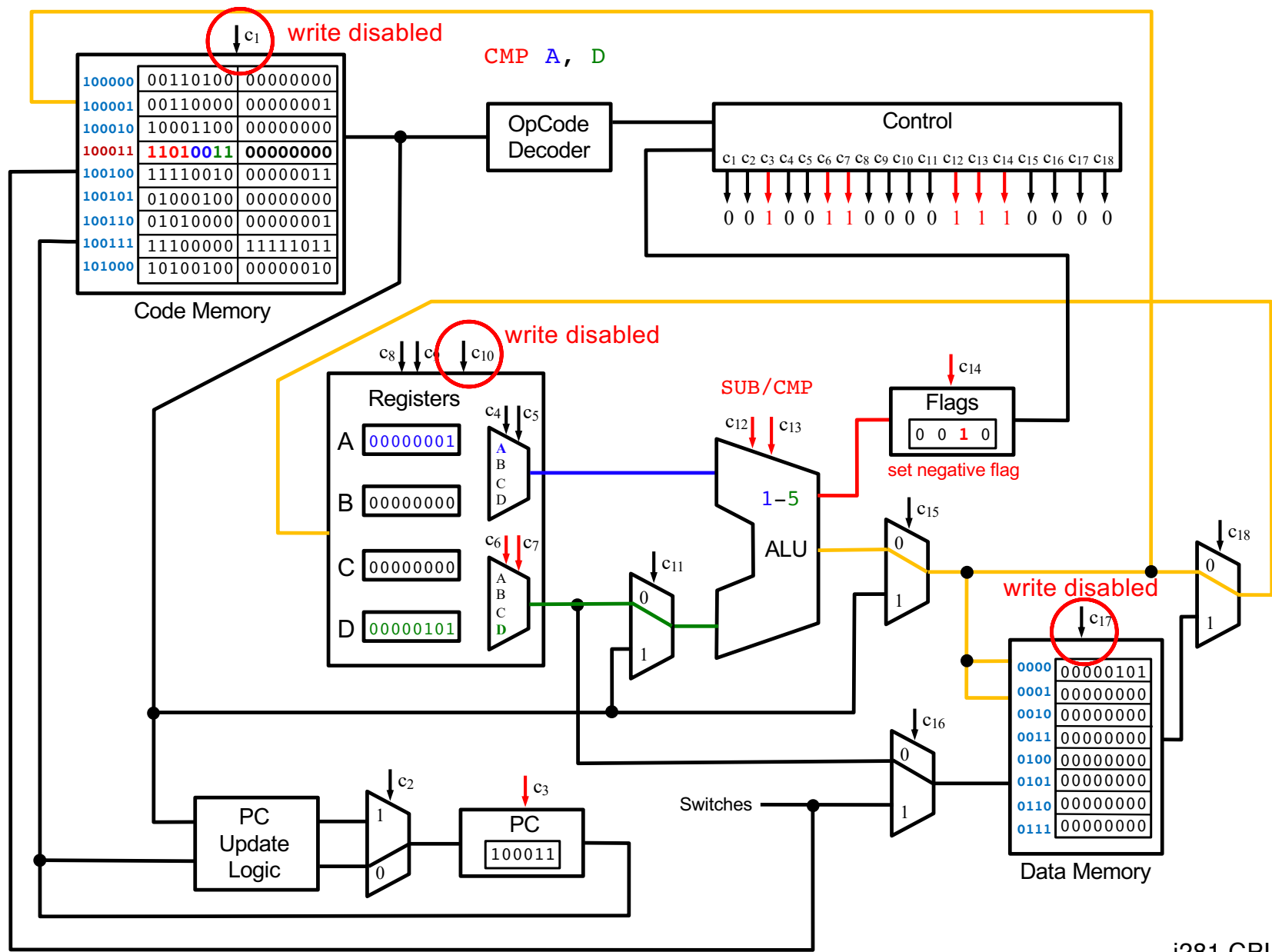
i281 CPU



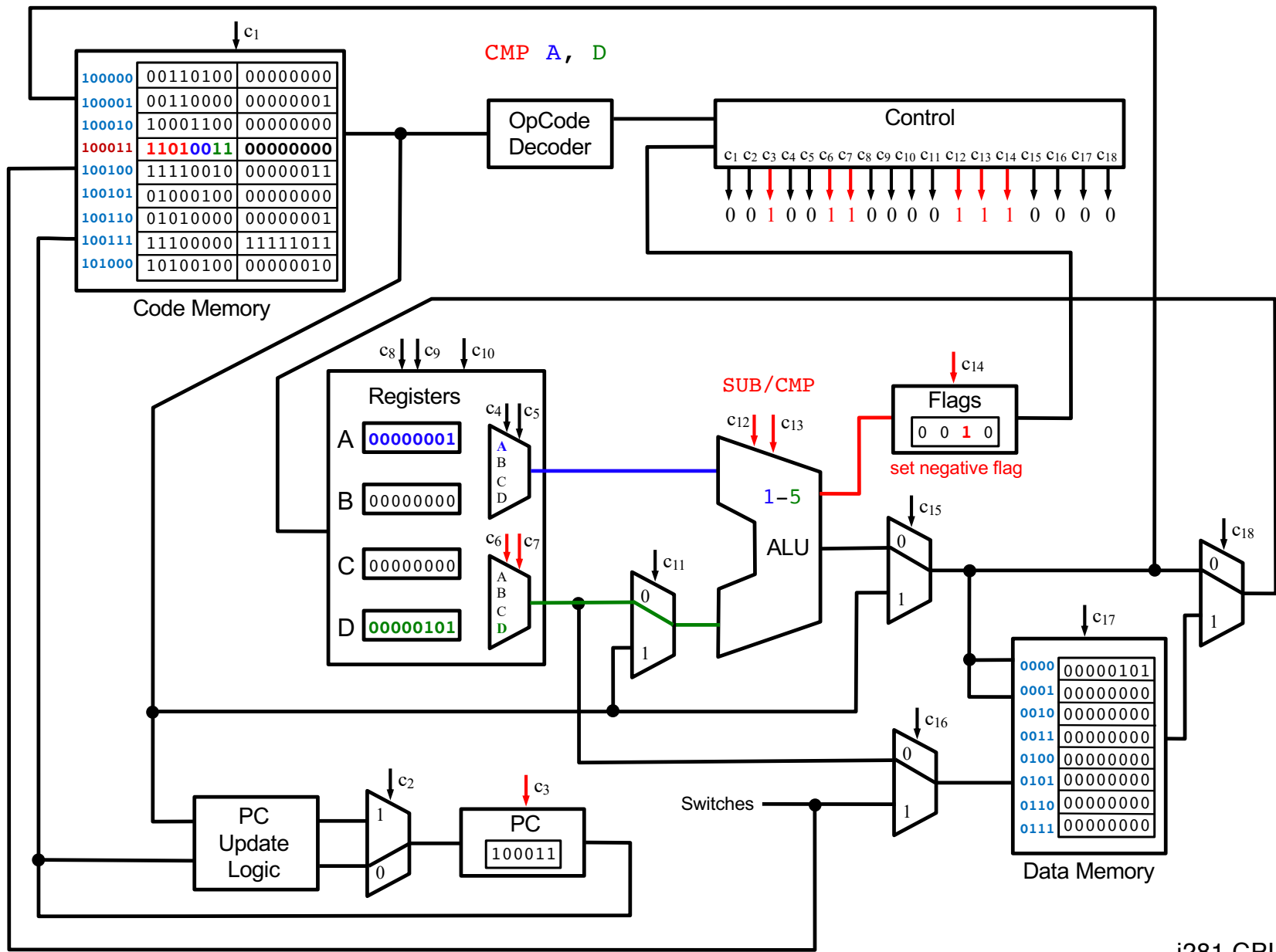
i281 CPU



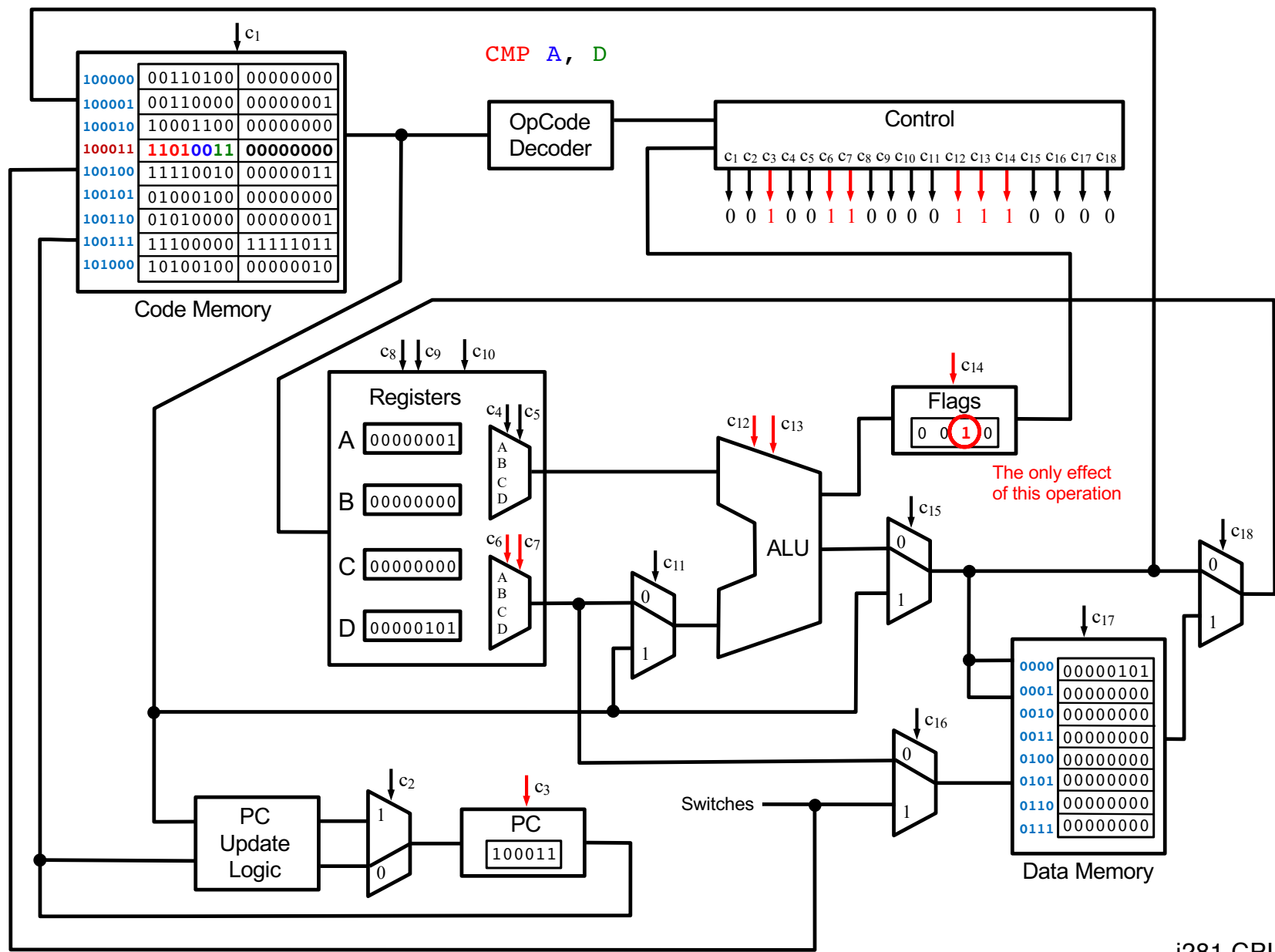
i281 CPU



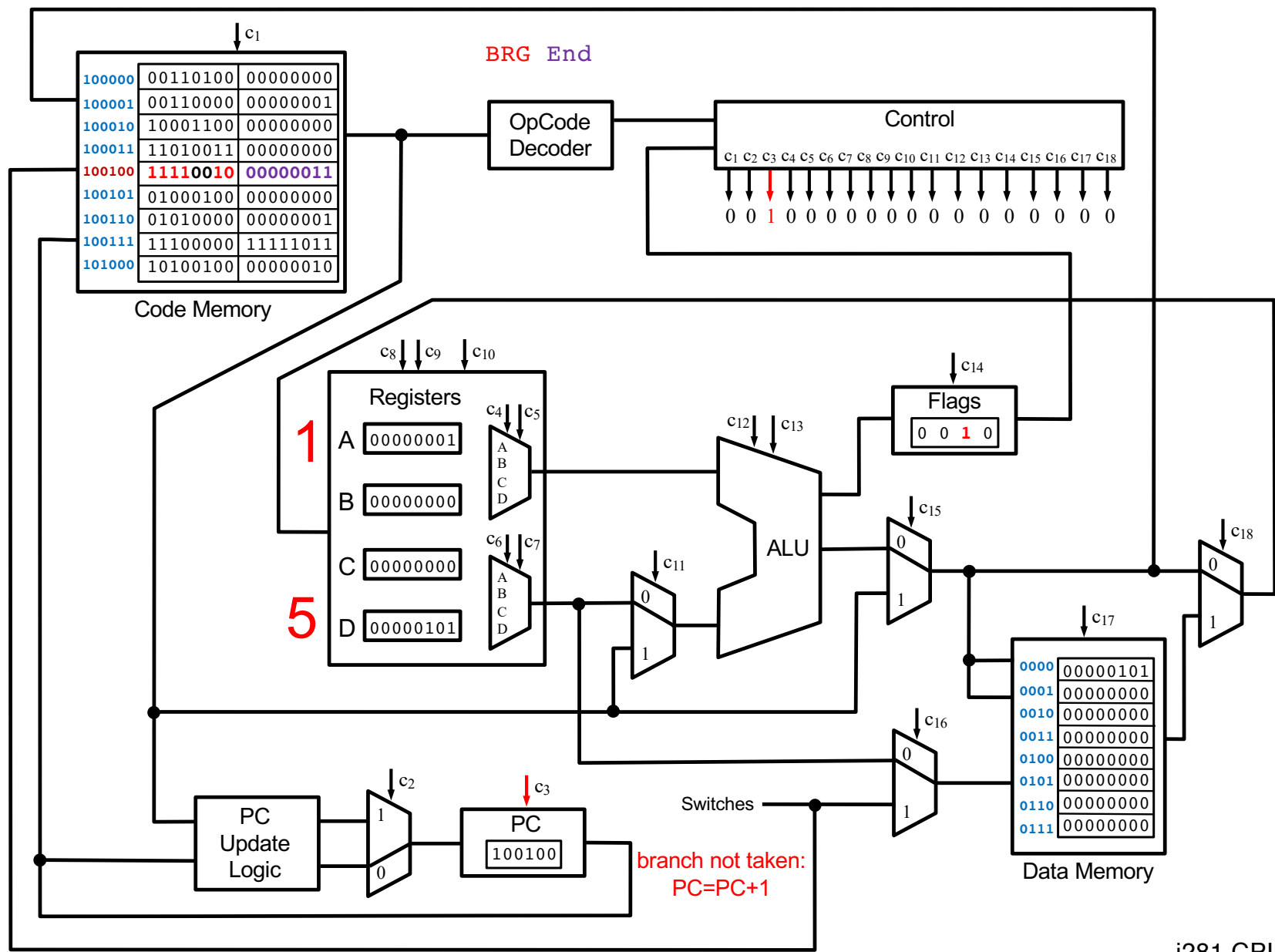
i281 CPU



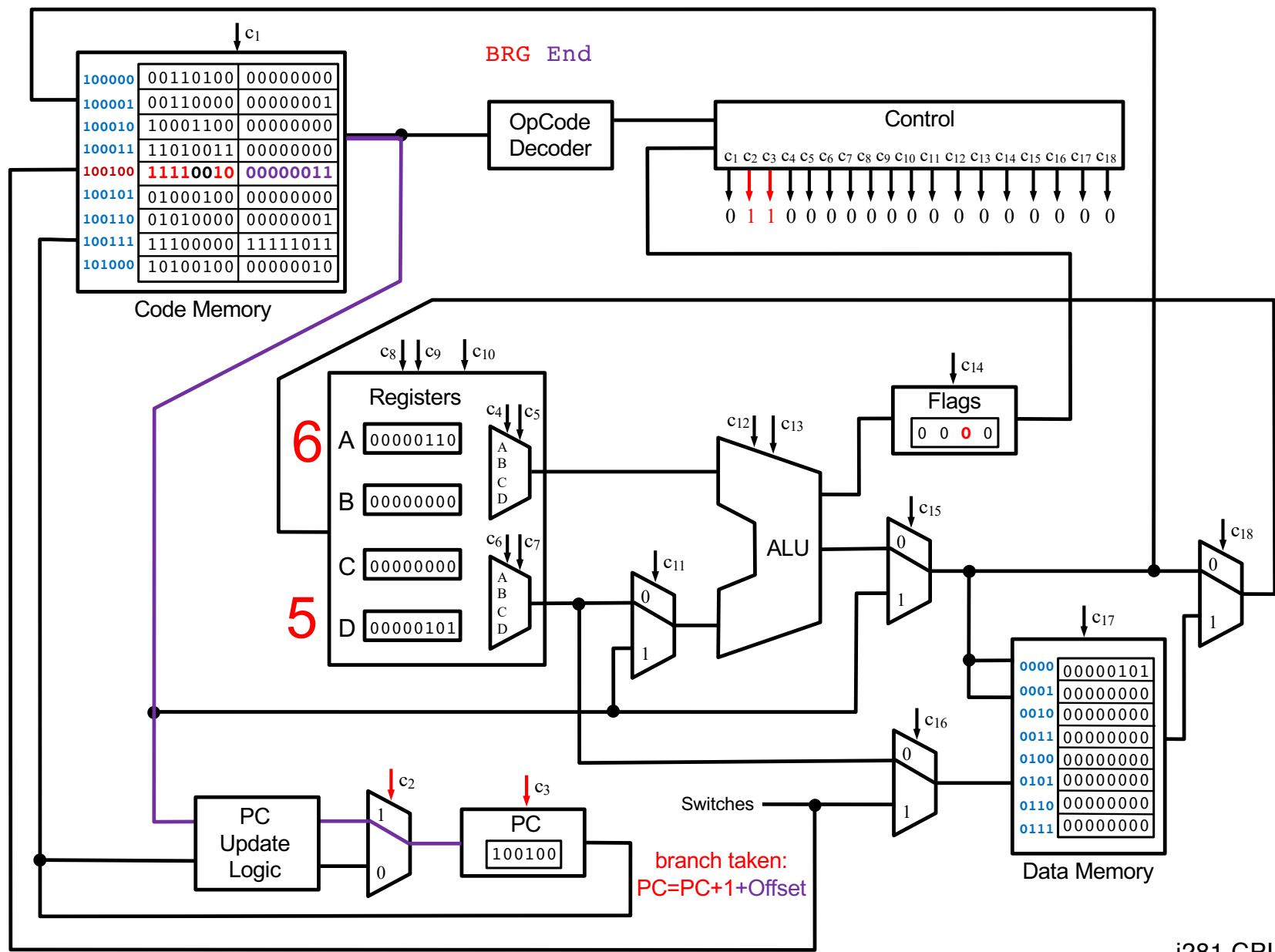
i281 CPU



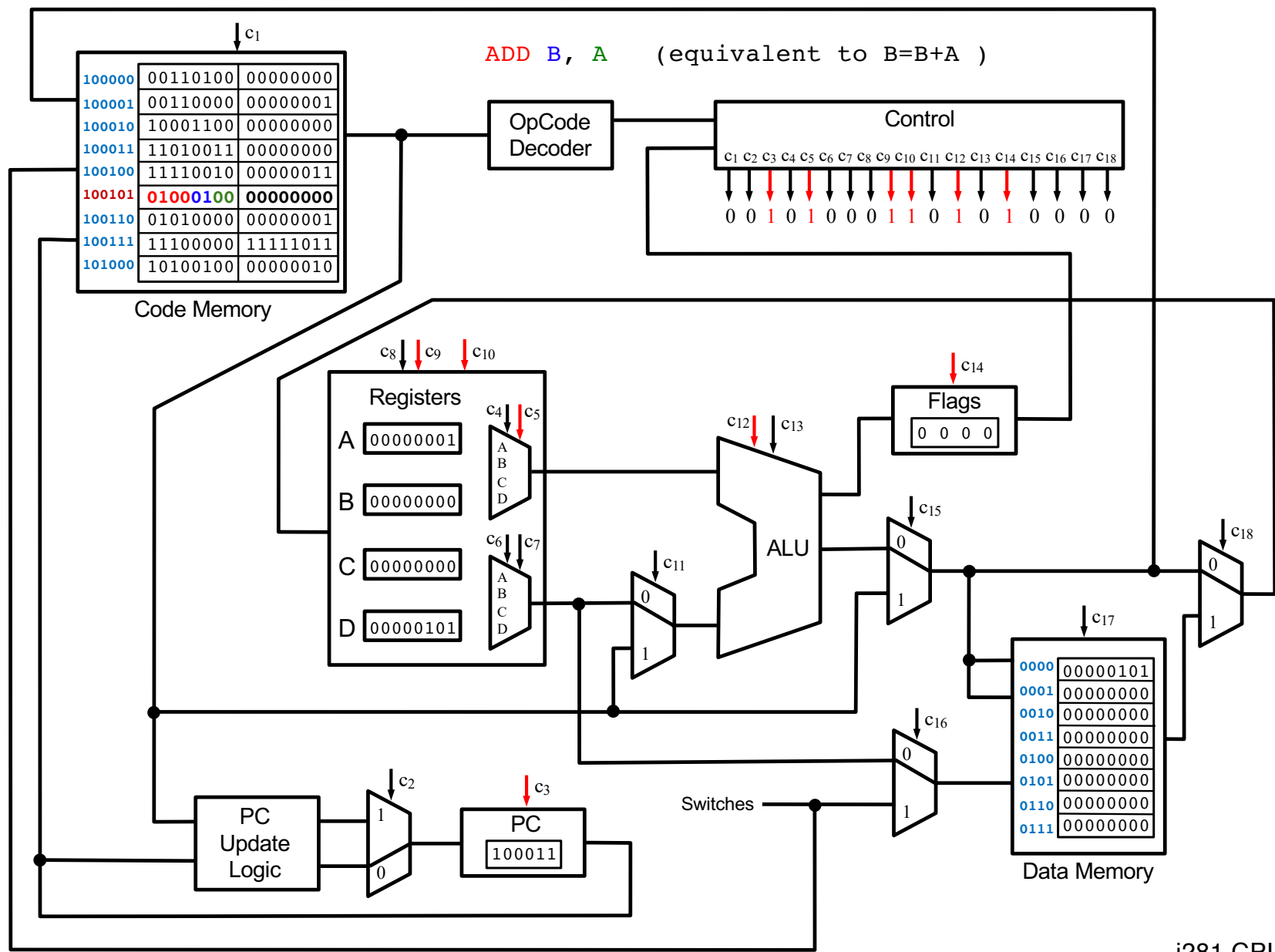
i281 CPU



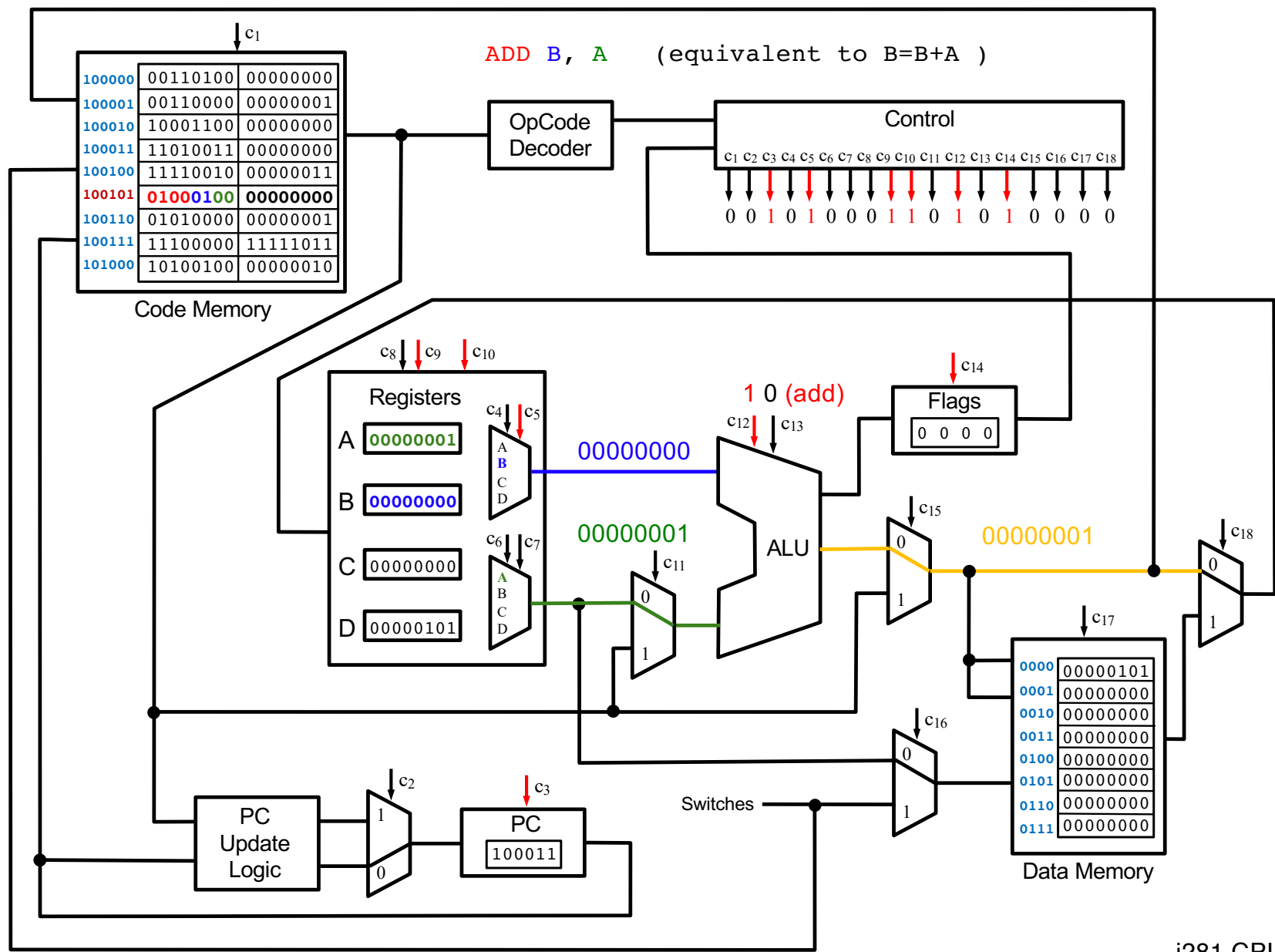
i281 CPU



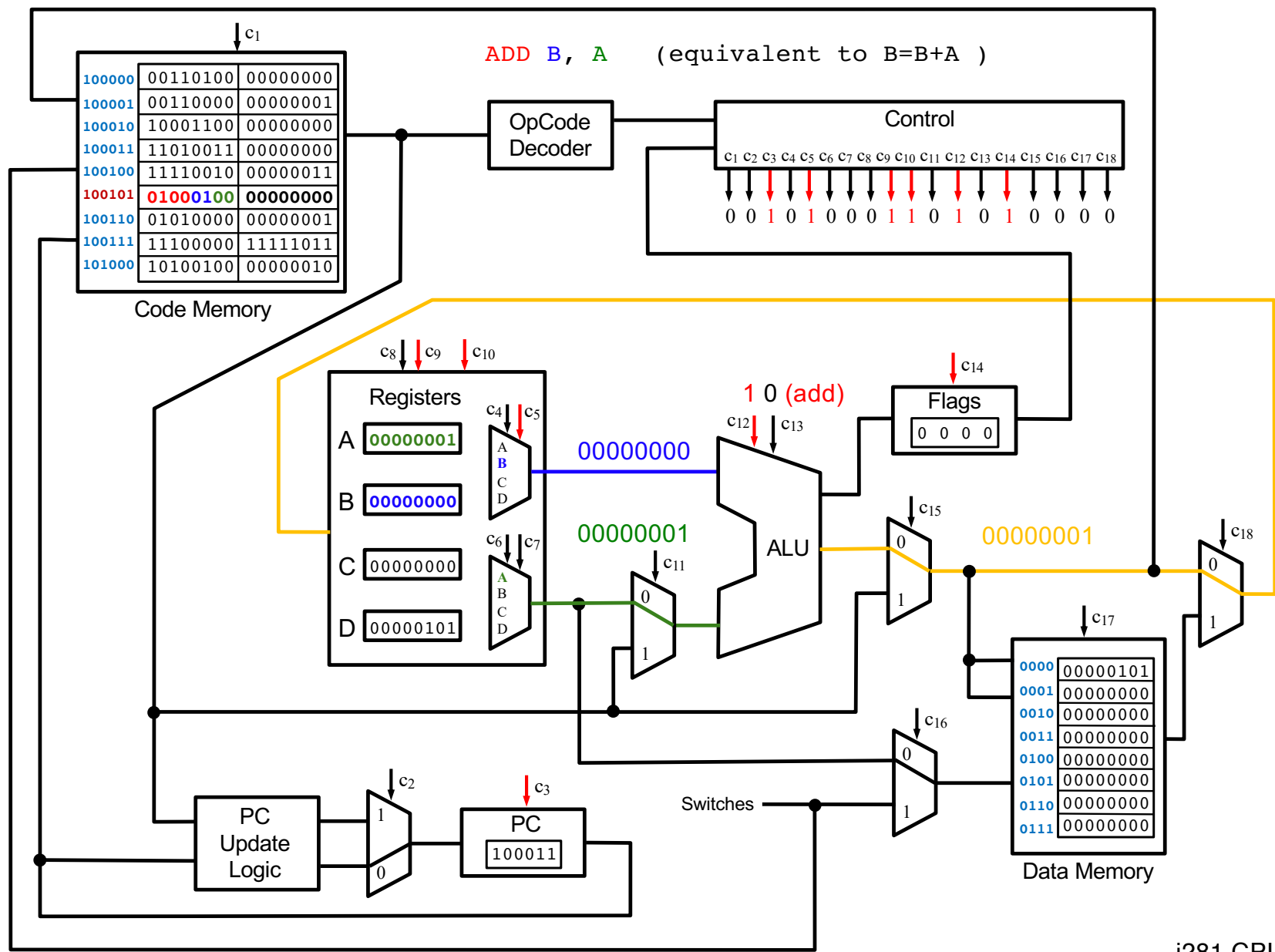
i281 CPU



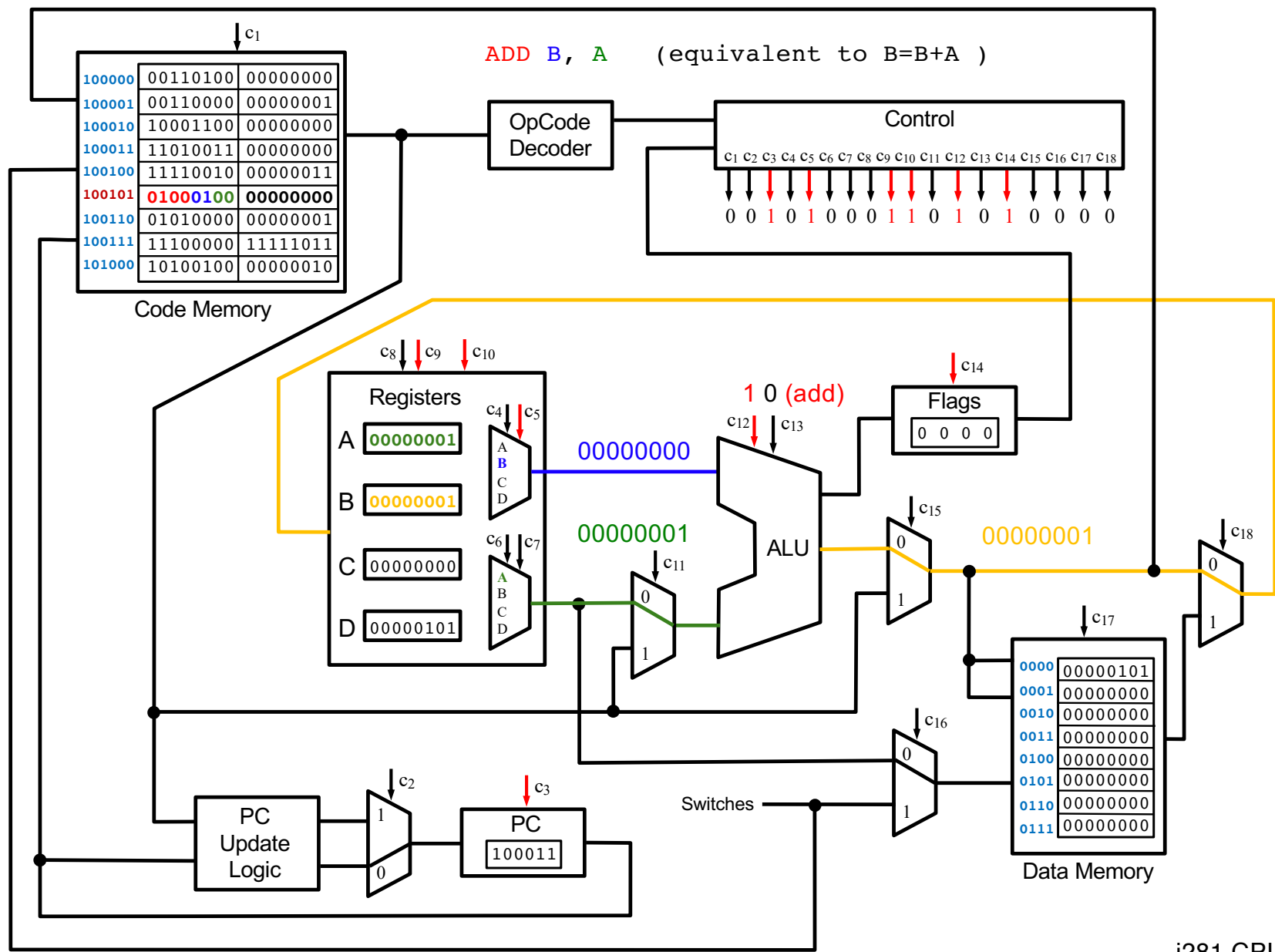
i281 CPU



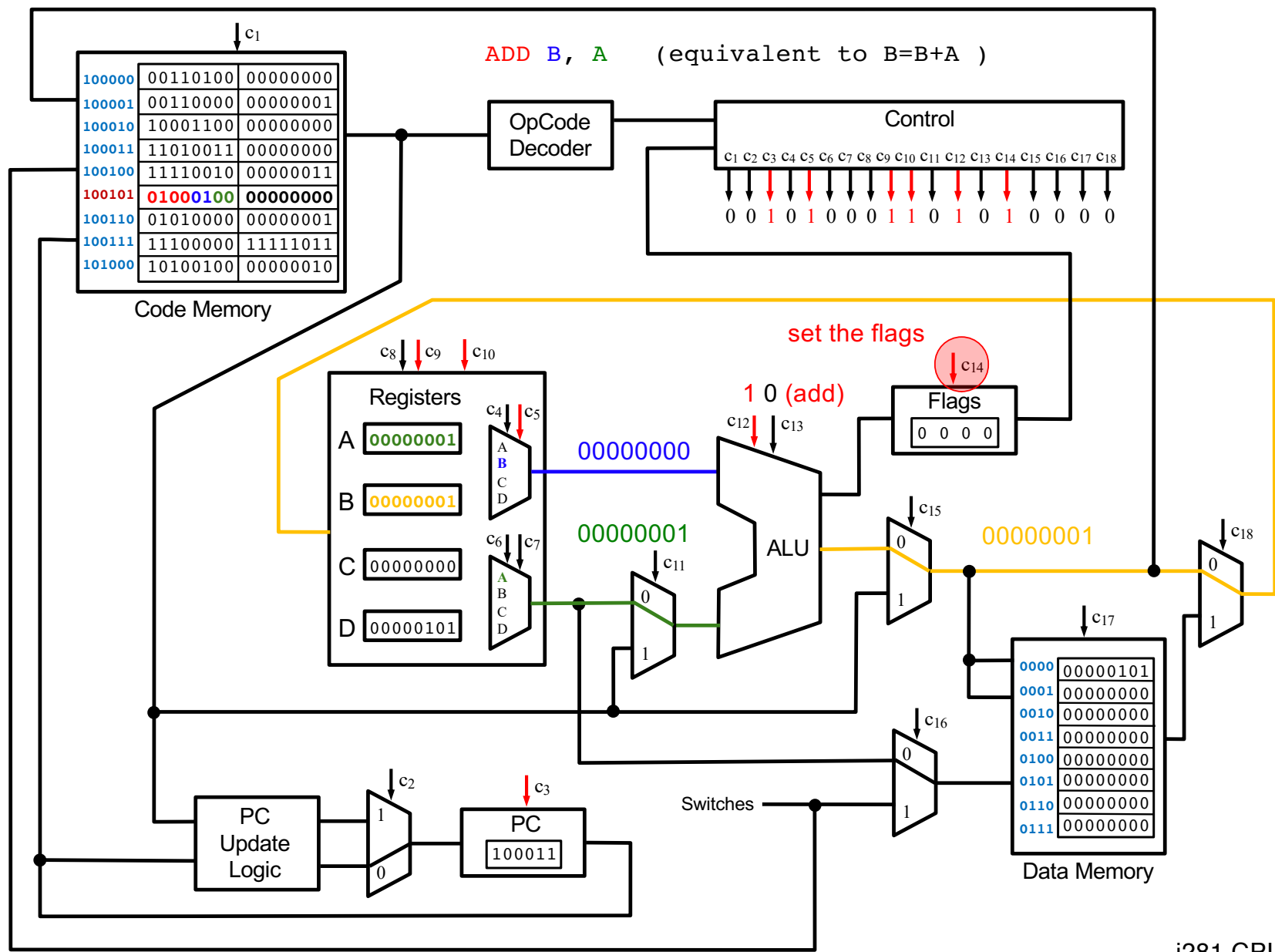
i281 CPU



i281 CPU



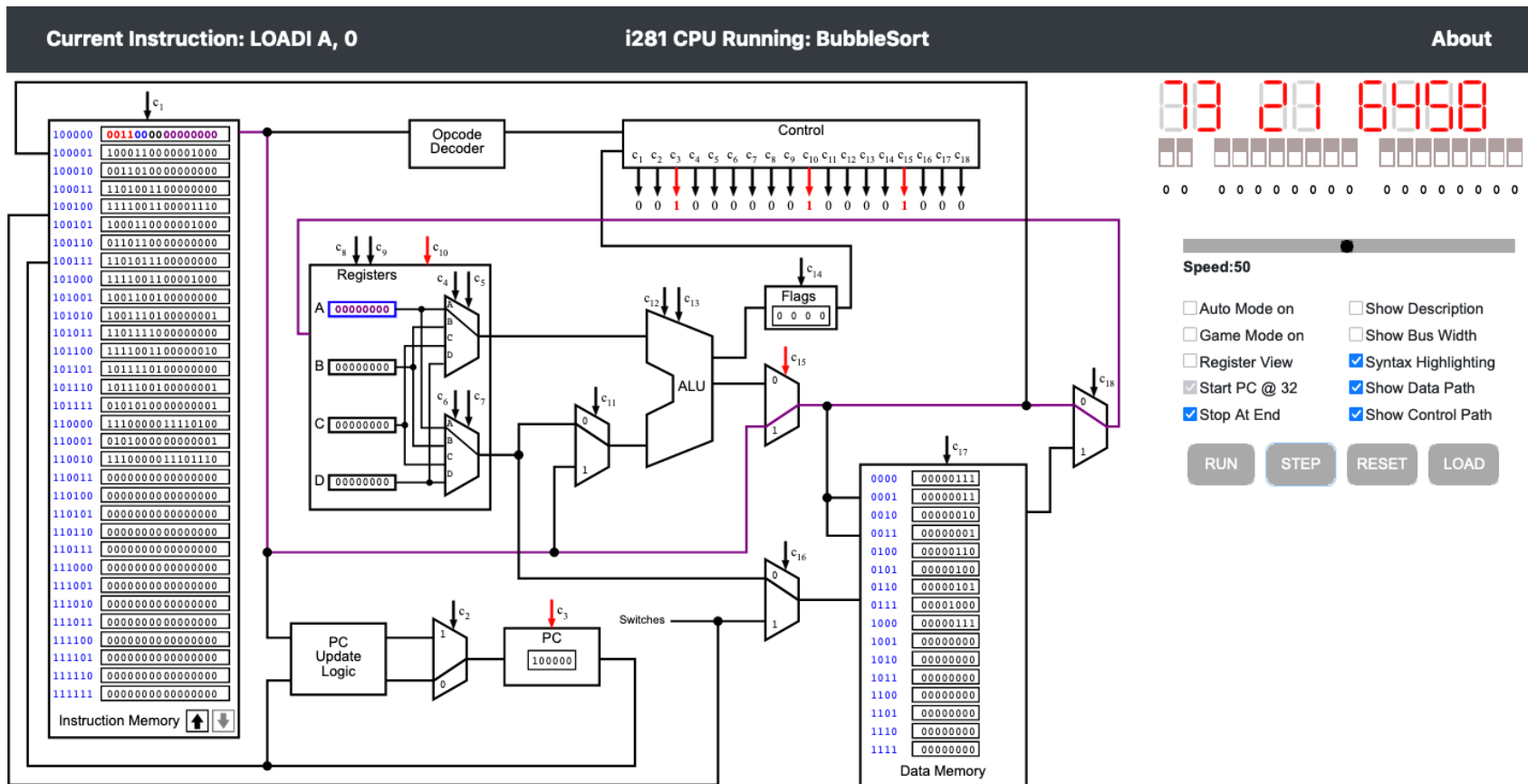
i281 CPU



i281 CPU

**For more examples
try the i281 simulator**

i281 Simulator



To try the simulator, go to the class web page and follow the link.

Questions?

THE END