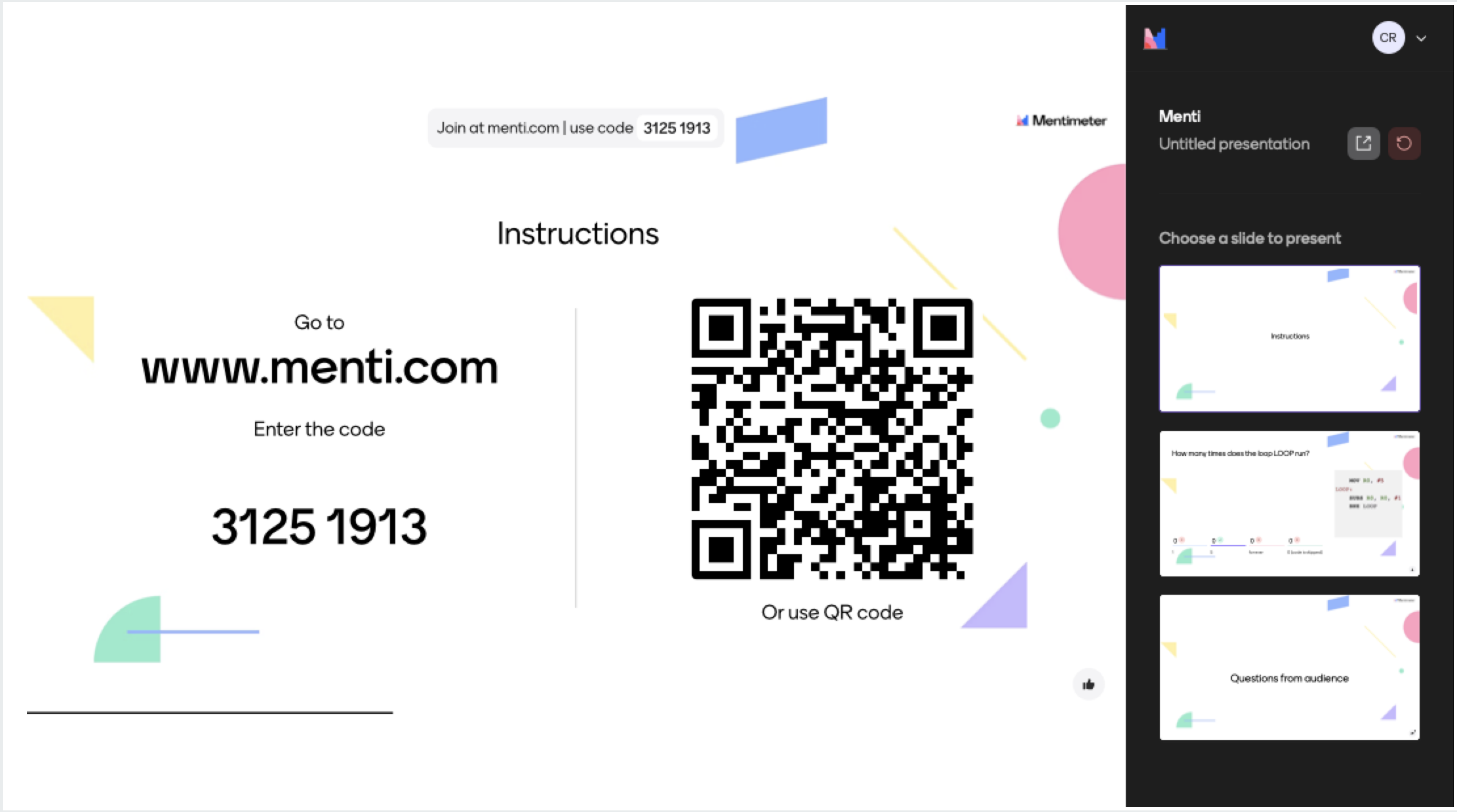


Join at menti.com | use code 3125 1913

Mentimeter

Instructions

Go to

www.menti.com

Enter the code

3125 1913

Or use QR code

Menti

Untitled presentation

Choose a slide to present

Instructions

How many times does the loop LOOP run?

```
    MOV R0, #5
LOOP:
    SUBS R0, R0, #1
    BNE LOOP
```

0 | 0 | 0 | 0
1 | 5 | forever | 0 (code is skipped)

Questions from audience

Revision questions

- What does the following instruction do?

```
LDR R2, [R1, #4]
```

Write the value in register R2, in address R1 + 4

- What is the difference between LDR and STR instructions?
LDR reads a value from memory, STR stores a value in memory

- What will be stored in memory at the address in R3 after executing the following code?

```
MOV R1, #10
MOV R2, #20
STR R1, [R3]
STR R2, [R3, #4]
```

R3[0] = 10, R3[1] = 20 (i.e., R3 + 4, assume R3 is int)*

ARM Assembly – Branching/Conditional Execution

Haris Rotsos

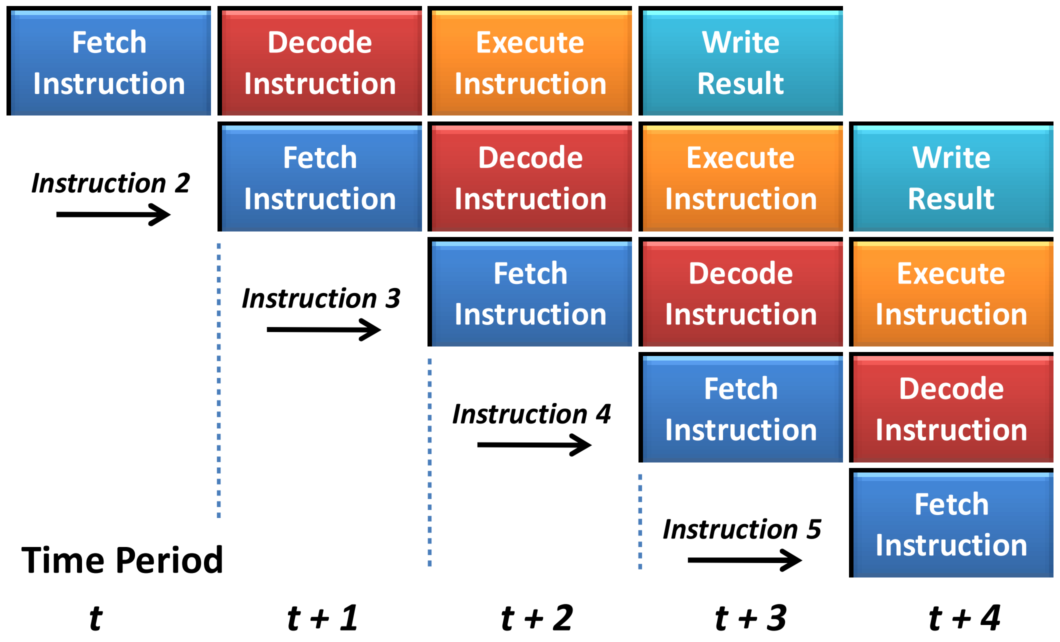
Introduction

- You have learned so far:
  - How to compile code
  - How to manipulate data
  - How to save/read data
- Our sample ASM programs are linear.
  - Usually code execution is influenced by data.
- Today: Controlling program execution in Assembly.
  - Branching
  - Conditional execution (if, while, for loop)

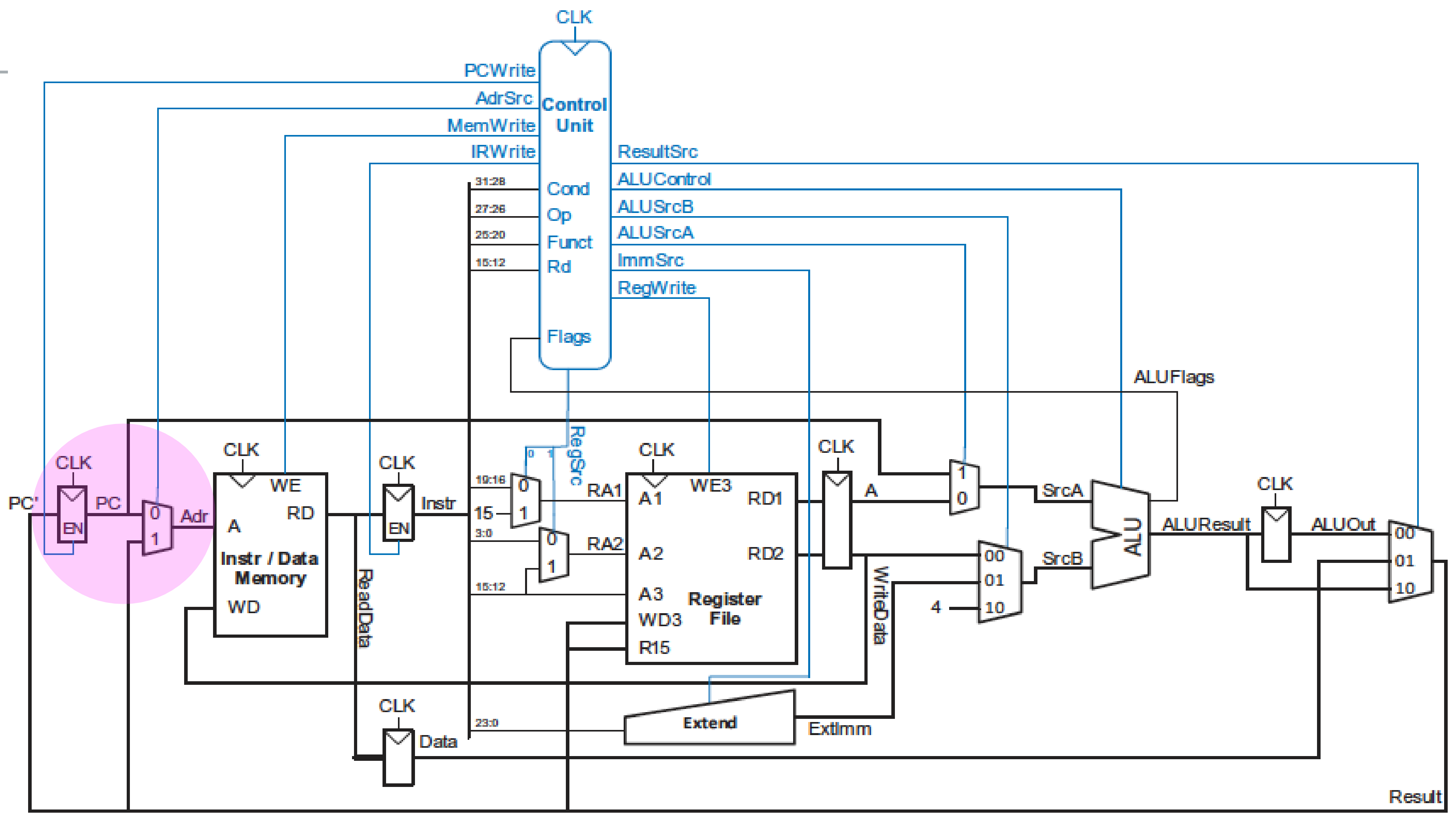
```c
#include<stdio.h>

int main(){
  int number=0;
  printf("Enter a number:");
  scanf("%d",&number);
  if(number%2==0){
    printf("%d is even",number);
  }
  return 0;
}
```

Pipelining

Pipelining is a widely-used way to exploit inherent **parallelism** inside the control unit to **speed up** the fetch-decode-execute cycle



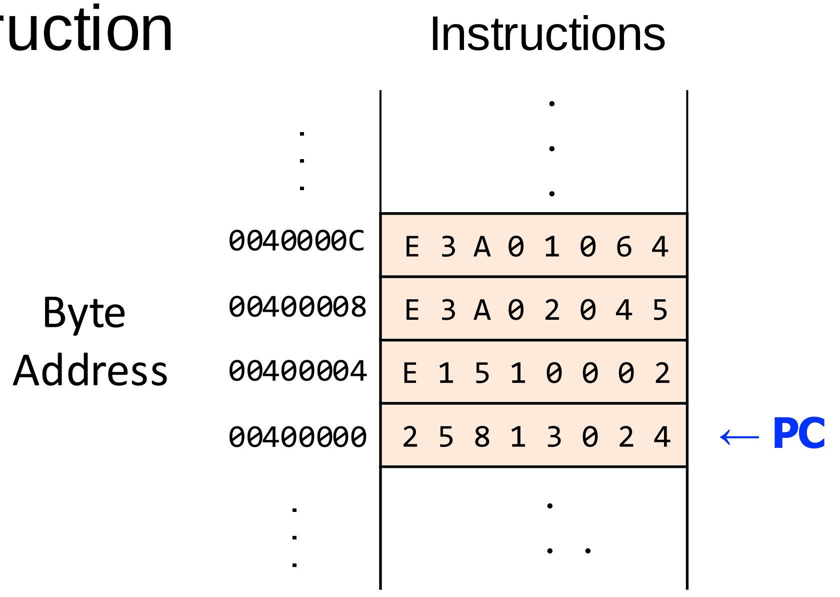
ARMv4 (Multi-Cycle) 32-bit



Program Counter (PC)

- The PC register contains the address of the next instruction to be executed
  - PC = PC + 4 during fetch to point to next instruction

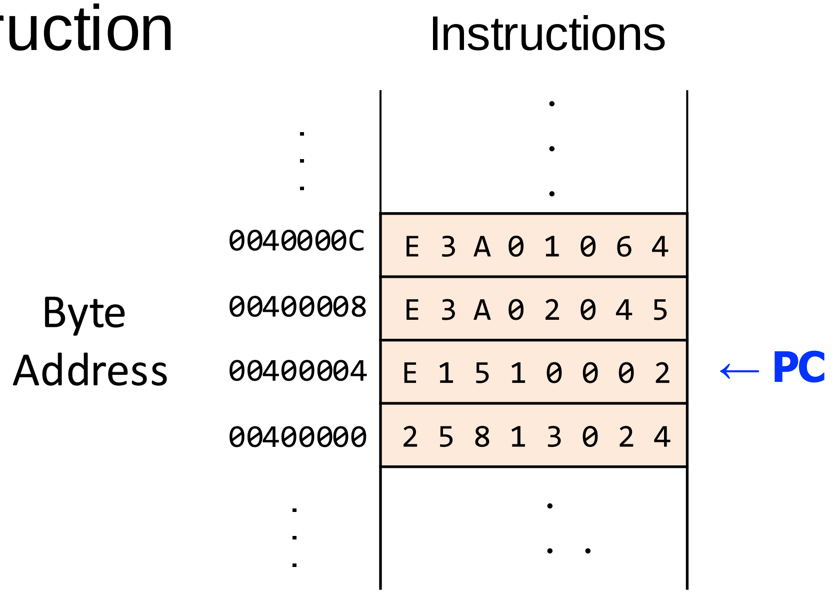
- PC = PC + 4 to execute the next
  - **sequential instruction** in memory



Program Counter (PC)

- The PC register contains the address of the next instruction to be executed
  - PC = PC + 4 during fetch to point to next instruction

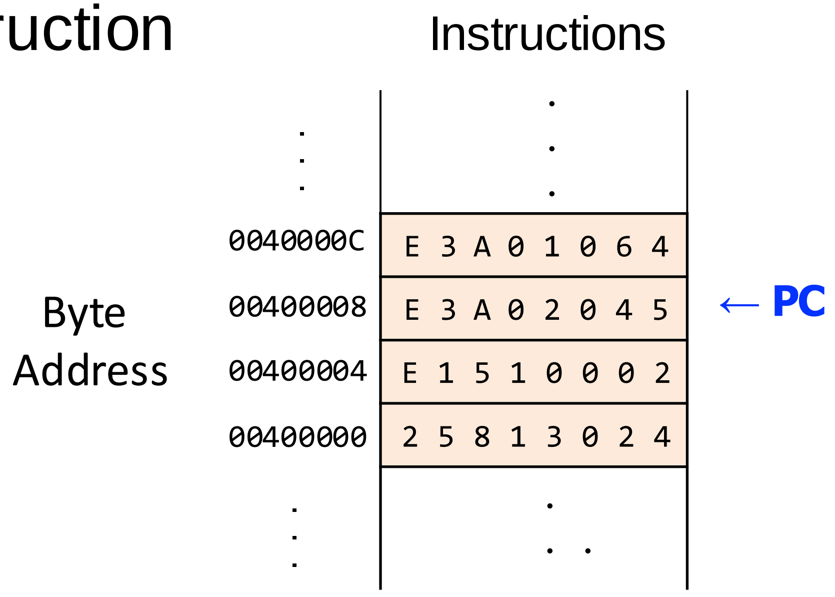
- PC = PC + 4 to execute the next
  - **sequential instruction** in memory



Program Counter (PC)

- The PC register contains the address of the next instruction to be executed
  - PC = PC + 4 during fetch to point to next instruction

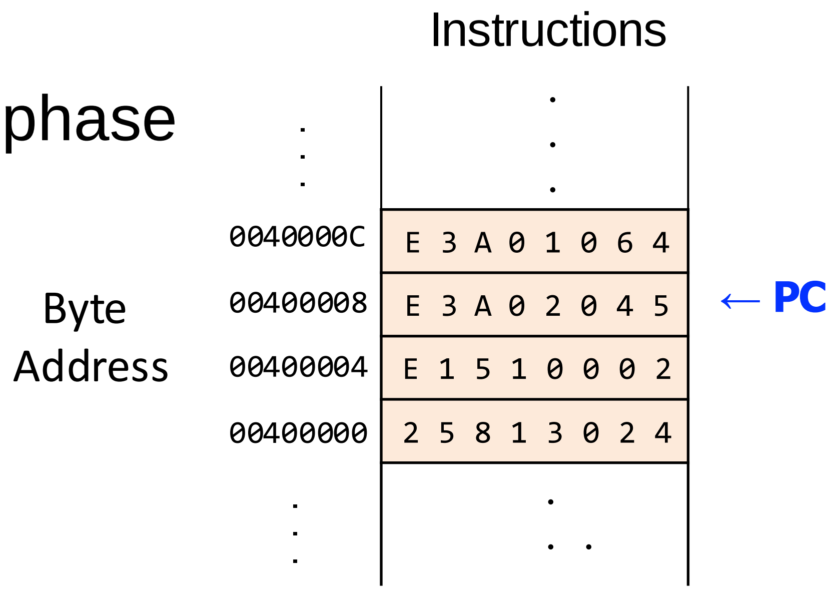
- PC = PC + 4 to execute the next
  - **sequential instruction** in memory



Branching Instruction and PC

- Branch instructions change the **PC** to point to a different instruction than the next sequential instruction in memory
    - Updated by a different address in the execute phase

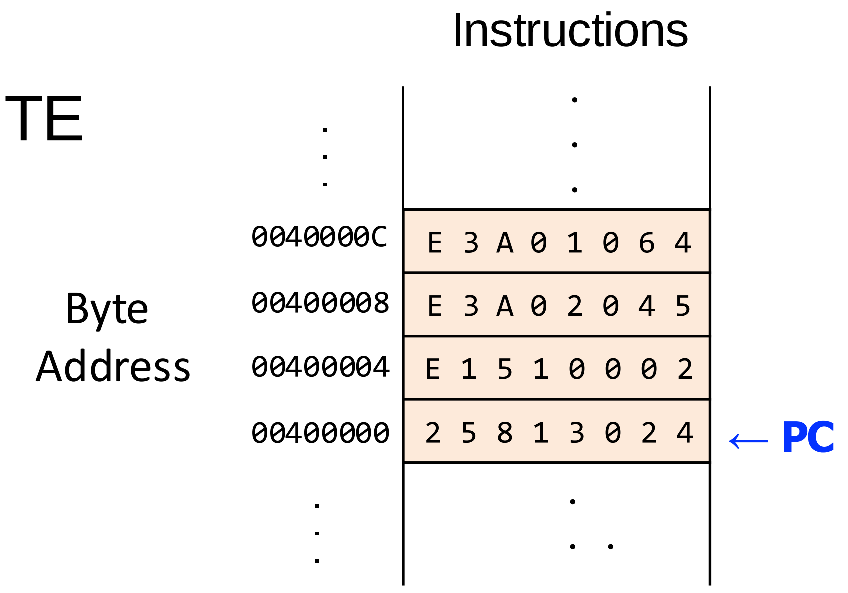
- PC = PC + 4 to execute the next
    - **sequential instruction** in memory



Branching Instruction and PC

- Branch instructions change the **PC** to point to a different instruction than the next sequential instruction in memory
  - Updated by a different address in the EXECUTE phase

- PC = PC + 4 to execute the next
  - **sequential instruction** in memory



Type of Branches

- Unconditional branching: always executes the target instruction

"branch" operator

```
b #100
b target_label
b r0
```

- Conditional branch: Perform a jump when a condition is true (more to come…)

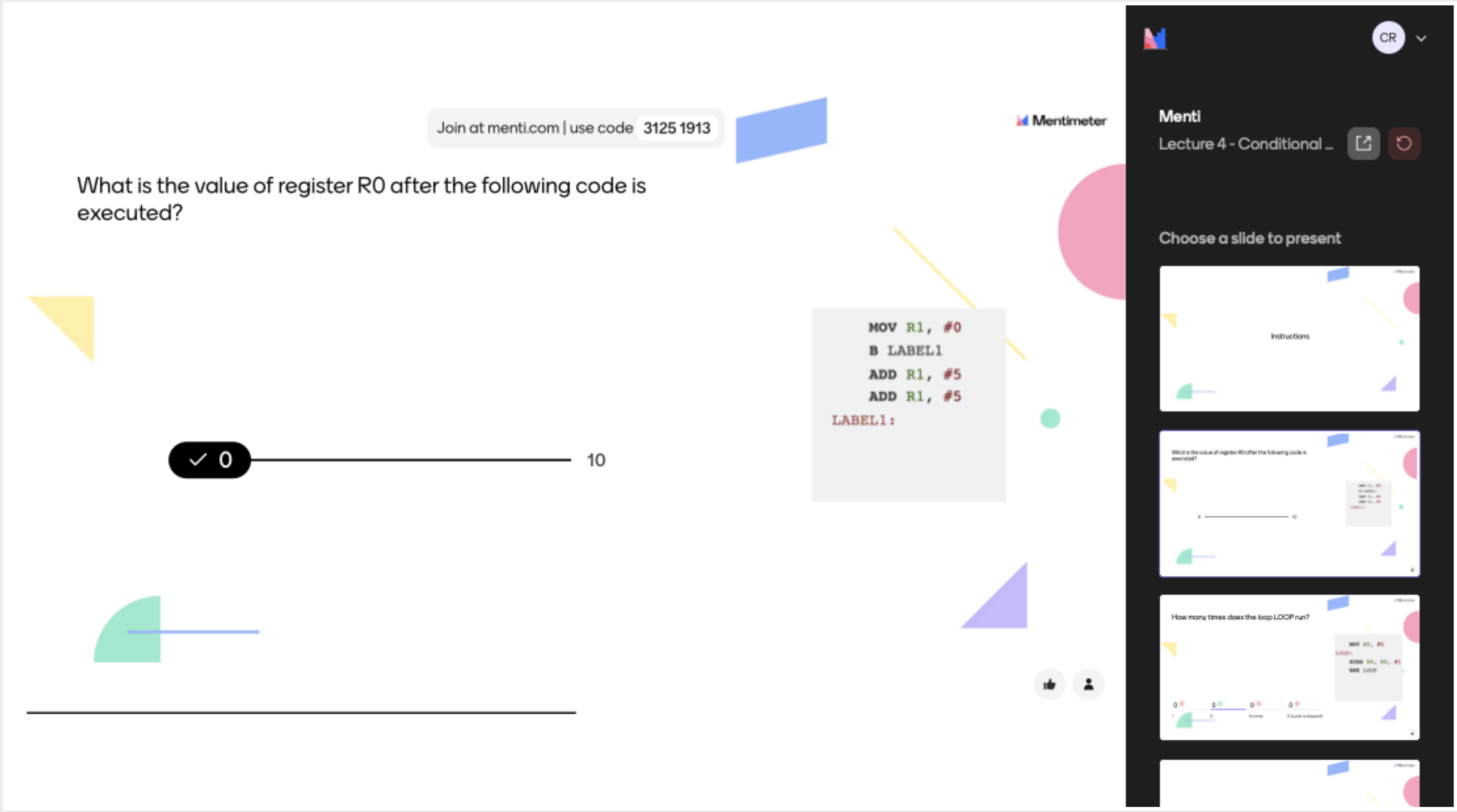
Label

- Labels are symbols that represent addresses.
  - The address given by a label is calculated during assembly.
  - Can point both to instructions or data.
  - Develops avoid manually calculating addresses.

- A reference to a label within the same section uses the program counter plus or minus an offset.
- This is called *program-relative addressing*.

Example: Unconditional Branch

```
    mov r1, #16
    mov r2, #16
    b   skip        <- +8 to pc to skip
    lsl r1, 2          next two instructions
    lsl r2, 2
skip:
    add r0, r1, r2
```



Join at menti.com | use code 3125 1913

Mentimeter

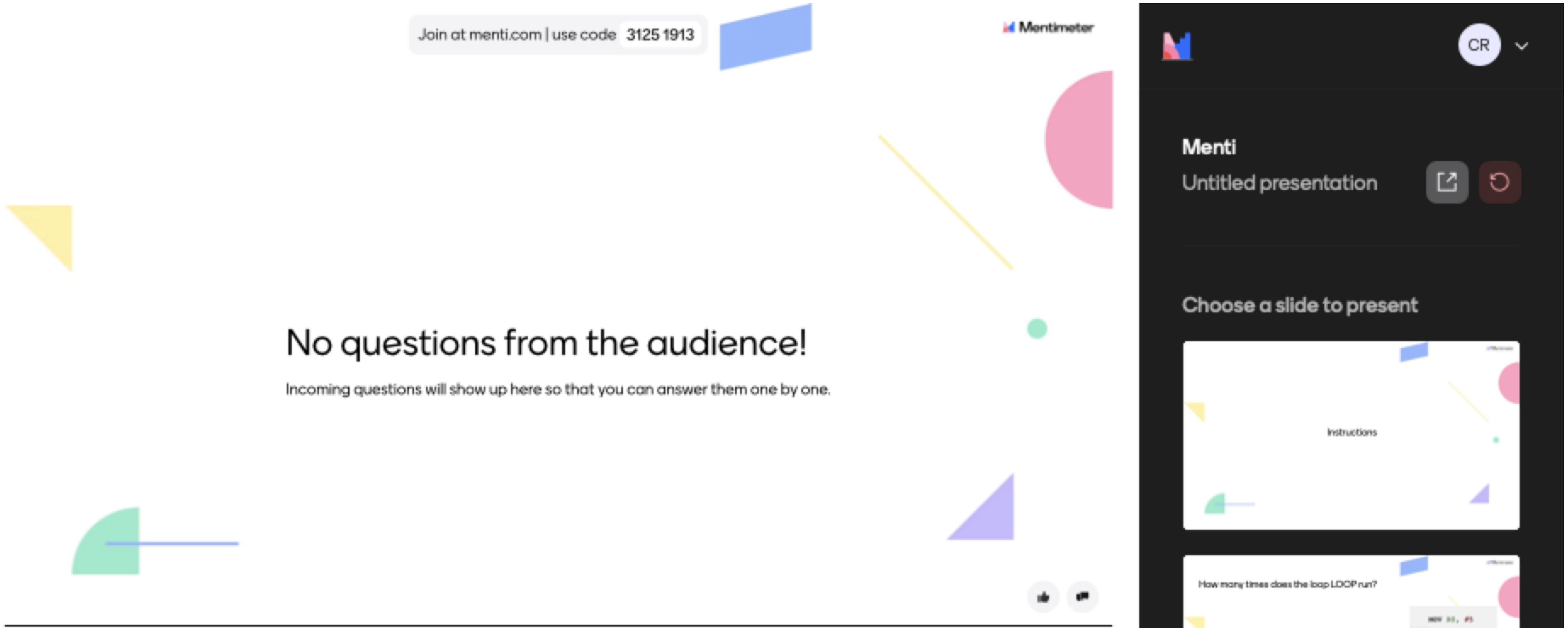
What is the value of register R0 after the following code is executed?

```
    MOV R1, #0
    B LABEL1
    ADD R1, #5
    ADD R1, #5
LABEL1:
```

✓ 0 —— 10

CR
Menti
Lecture 4 - Conditional ...
Choose a slide to present
Instructions
What is the value of register R0 after the following code is executed?
How many times does the loop LOOP run?

Questions?


Join at menti.com | use code 3125 1913

No questions from the audience!

Incoming questions will show up here so that you can answer them one by one.

Menti

Untitled presentation

Choose a slide to present

Instructions

How many times does the loop LOOP run?

```
MOV R0, #5
```

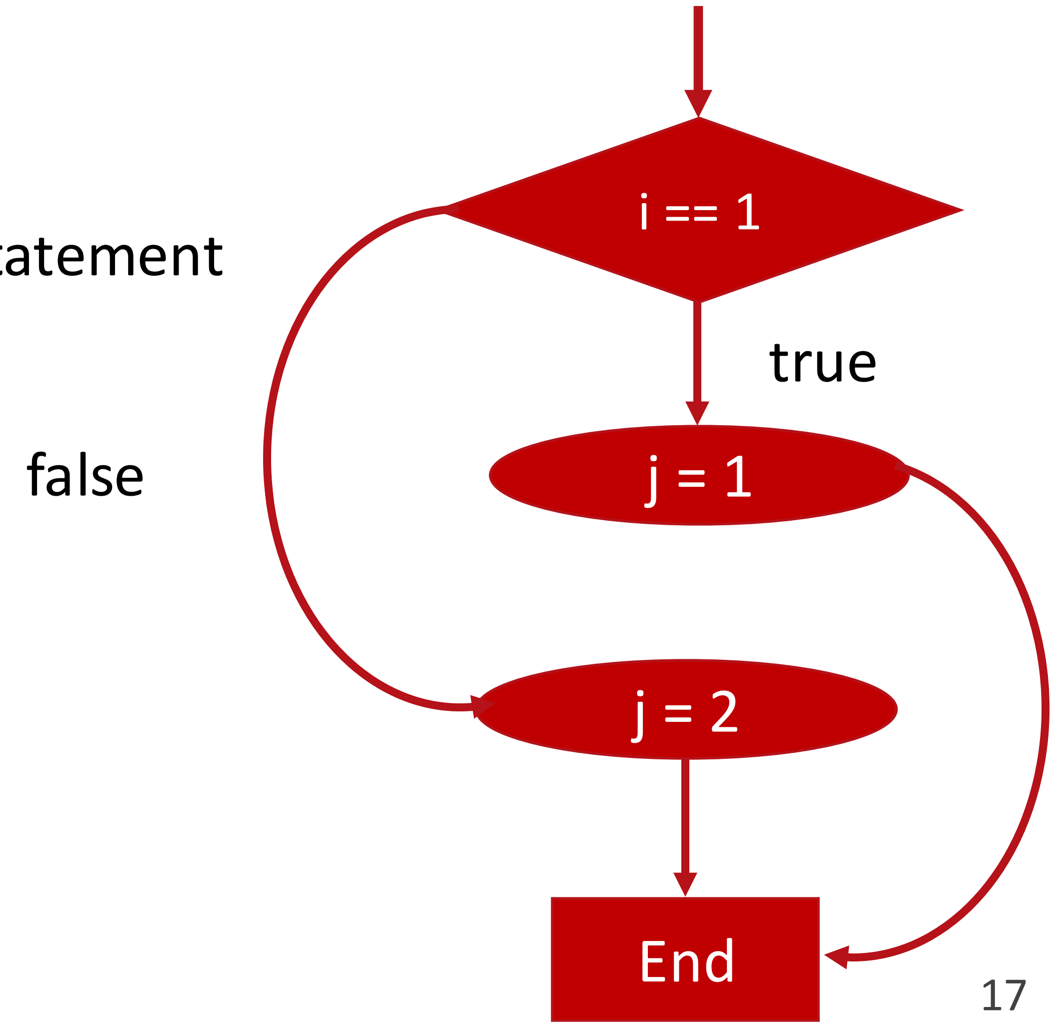
Conditional code – if statements

C code example

```
if (i == 1) {
  j = 1;
} else {
  j = 2;
}
```

C code example – goto statement

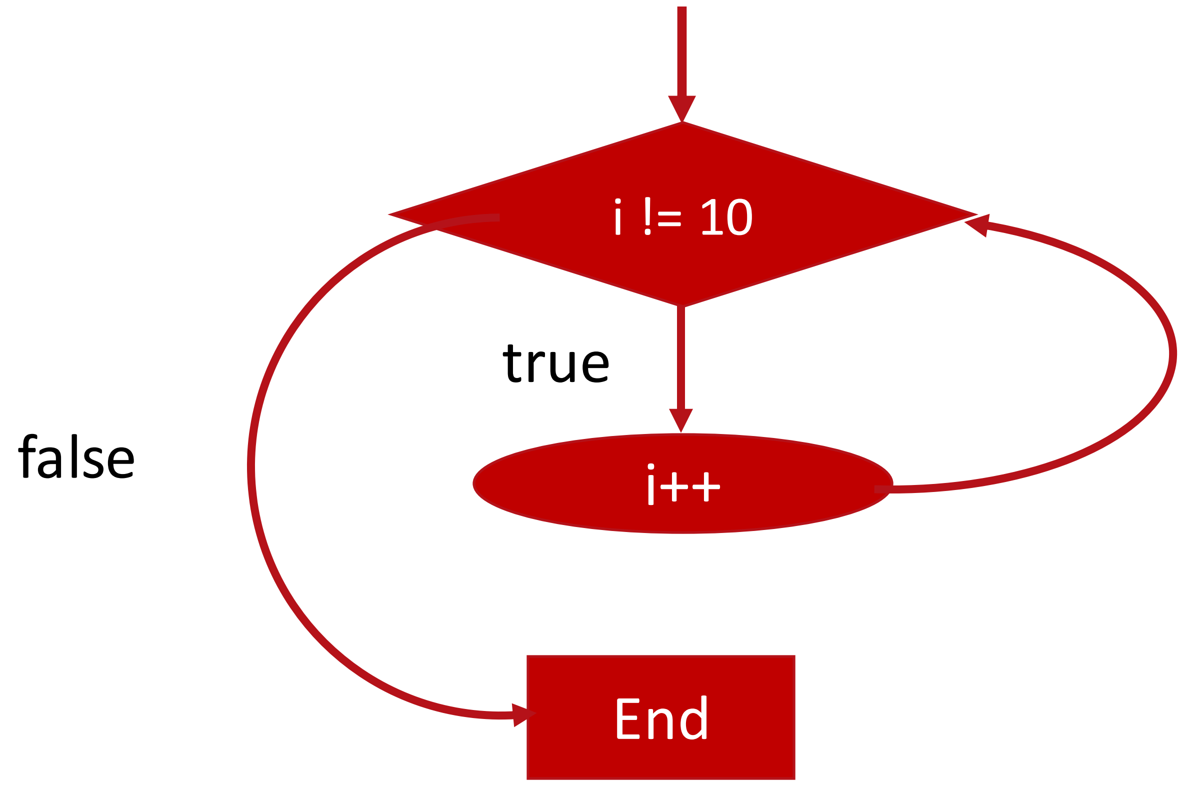
```
if (i == 1)
   goto notone;
j = 2;
goto exit;

notone:
j = 1;
exit:
```



Conditional code – loop statements

C code example

```
while (i != 10) {
  i++;
}
```



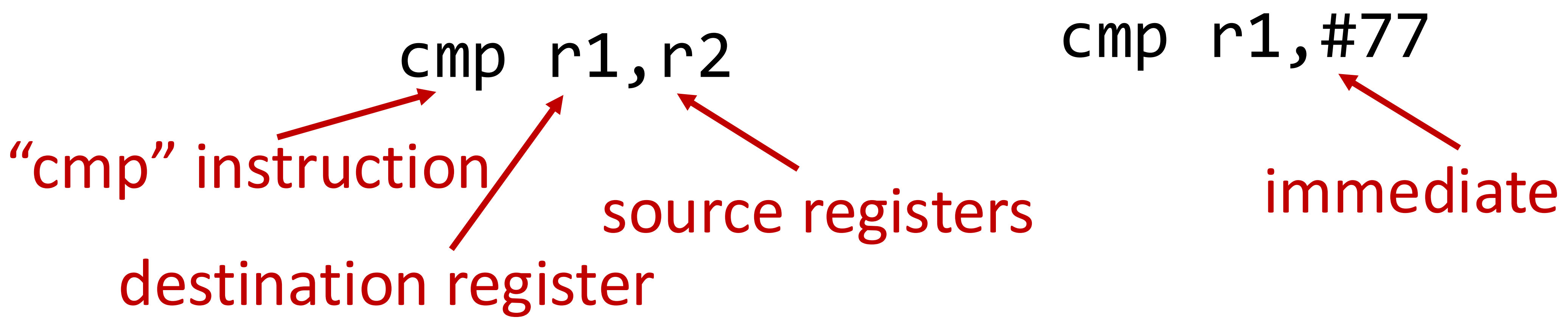
Current Program Status Register (CPSR)

- CPSR is an ARM register that records the state of the program.
  - Arithmetic instructions will affect its value every time.
- **N bit - "negative flag”:** instruction result was negative.
- **Z bit -"zero flag”:** instruction result was zero.
- **C bit - "carry flag”:** Instruction causes a carry-out or borrow.
- **V bit - "overflow flag”:** Instruction produces an overflow in 2's complement numbers.

31	30	29	28		7	6	5	4 0
N	Z	C	V	…	I	F	T	Mode

CoMPare

- Perform a comparison (substraction), updates the CPSR and discards the result



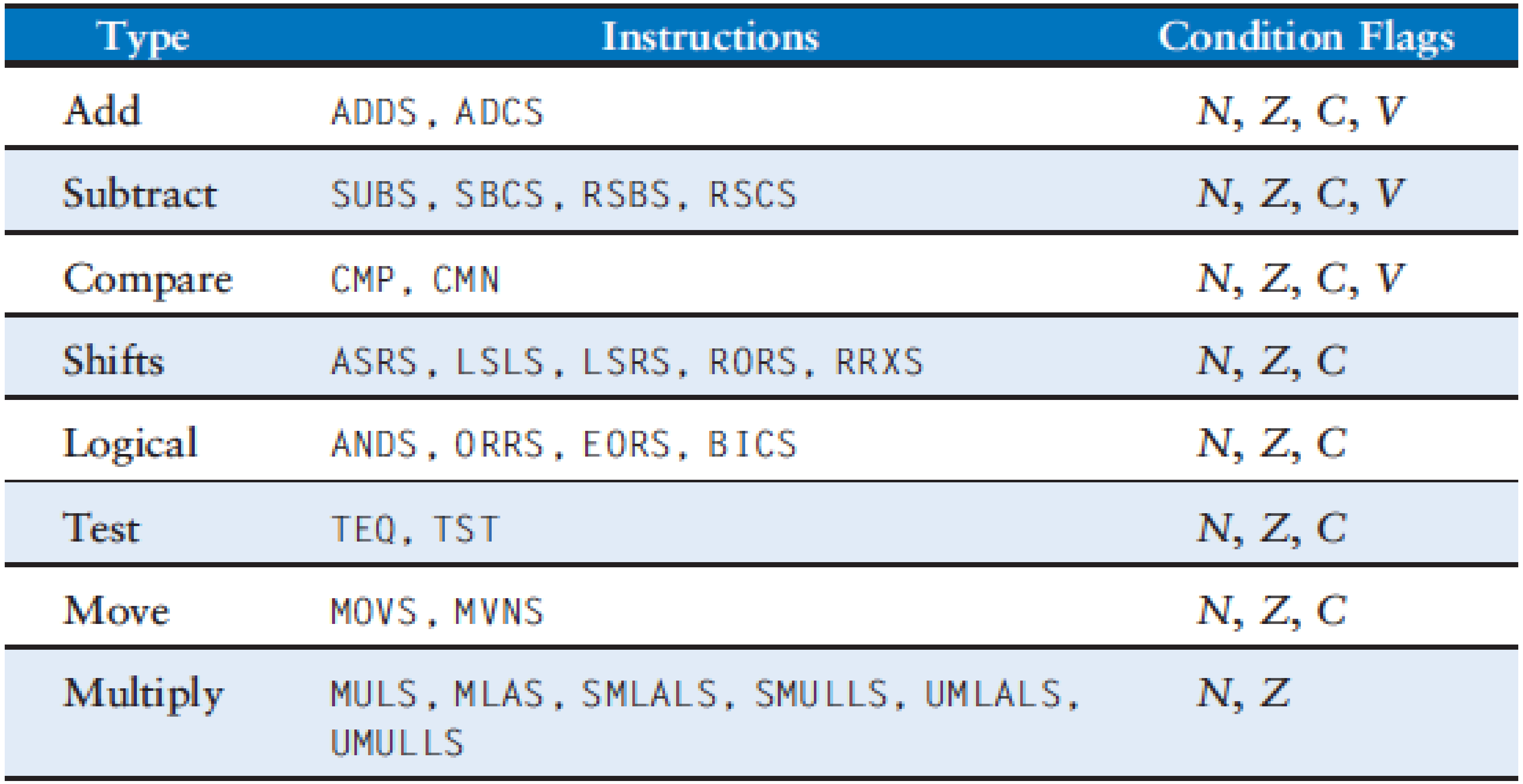
```
cmp r1,r2
```

```
cmp r1,#77
```

Compare the values of r2 in r1

Compare r1 to the value 77

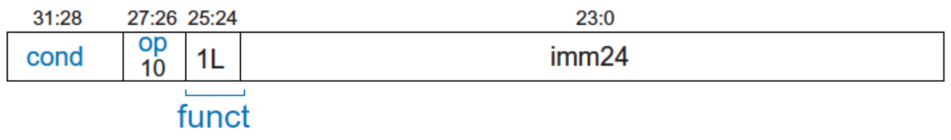
CPSR and arithmetic instructions



Type	Instructions	Condition Flags
Add	`ADDS, ADCS`	N, Z, C, V
Subtract	`SUBS, SBCS, RSBS, RSCS`	N, Z, C, V
Compare	`CMP, CMN`	N, Z, C, V
Shifts	`ASRS, LSLS, LSRS, RORS, RRXS`	N, Z, C
Logical	`ANDS, ORRS, EORS, BICS`	N, Z, C
Test	`TEQ, TST`	N, Z, C
Move	`MOVS, MVNS`	N, Z, C
Multiply	`MULS, MLAS, SMLALS, SMULLS, UMLALS, UMULLS`	N, Z

Conditional Branching

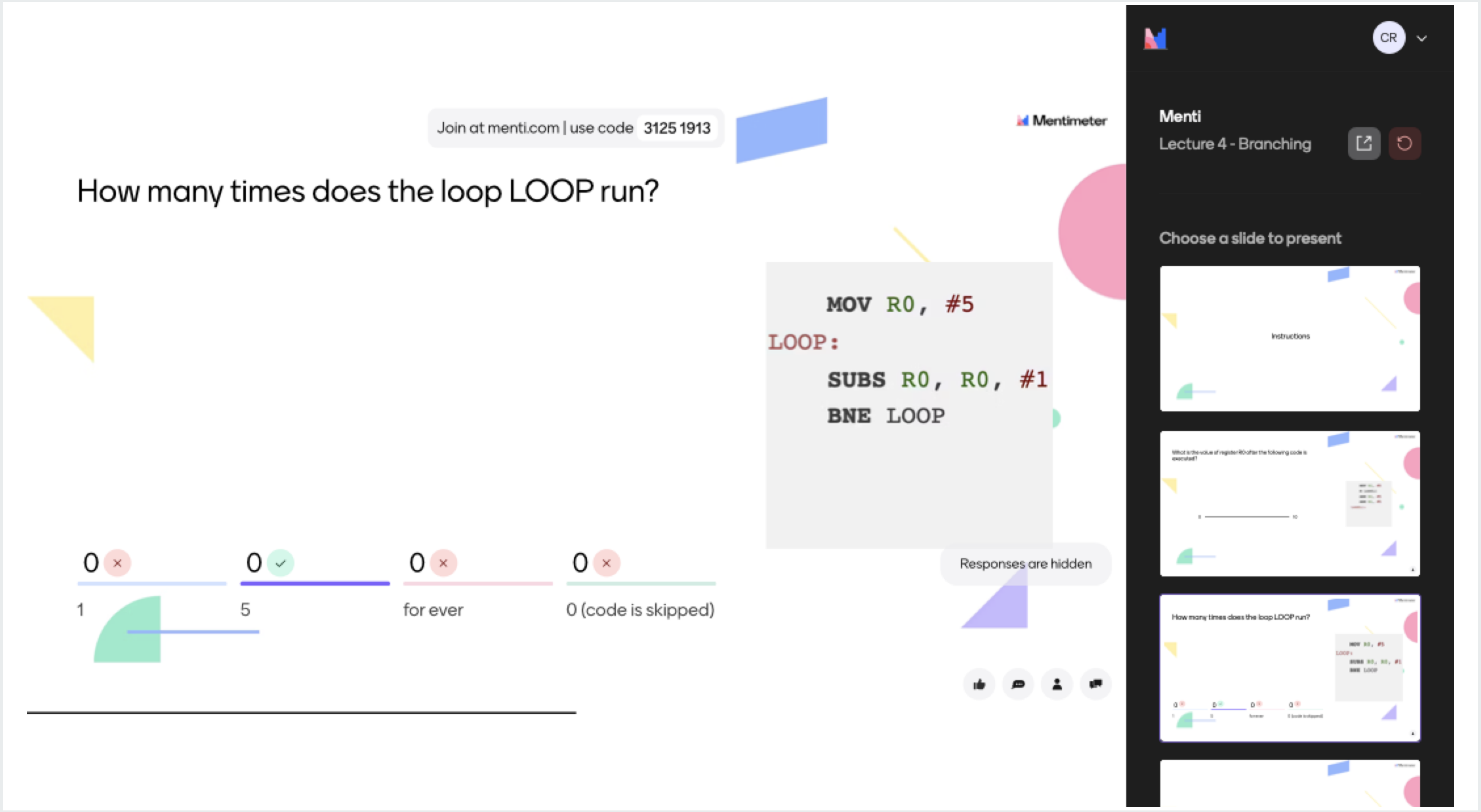
- Branch operations can also use the condition code in order to branch under certain conditions.
  - Beq: branch if equal
  - Bne: branch if not equal
  - Blt: branch if less than.
  - …
- Branch instructions use a single 24-bit signed immediate operand,
- The remaining 24-bit two's complement imm24 field is used to specify an instruction address relative to PC + 8.



Condition Branching

- Branching instruction variation: BEQ, BNE,…
- Their execution depends on the CPSR condition bits.
  - If condition is *false*, then branch is skipped.
- You can construct any branch condition with the mnemonics in the table.

cond	Mnemonic	Name	CondEx
0000	EQ	Equal	Z
0001	NE	Not equal	$\bar{Z}$
0010	CS / HS	Carry set / Unsigned higher or same	C
0011	CC / LO	Carry clear / Unsigned lower	$\bar{C}$
0100	MI	Minus / Negative	N
0101	PL	Plus / Positive of zero	$\bar{N}$
0110	VS	Overflow / Overflow set	V
0111	VC	No overflow / Overflow clear	$\bar{V}$
1000	HI	Unsigned higher	$\bar{Z}C$
1001	LS	Unsigned lower or same	$Z\ OR\ \bar{C}$
1010	GE	Signed greater than or equal	$\overline{N \oplus V}$
1011	LT	Signed less than	$N \oplus V$
1100	GT	Signed greater than	$\bar{Z}(\overline{N \oplus V})$
1101	LE	Signed less than or equal	$Z\ OR\ (N \oplus V)$
1110	AL (or none)	Always / unconditional	ignored



Join at menti.com | use code 3125 1913

Mentimeter

How many times does the loop LOOP run?

```
    MOV R0, #5
LOOP:
    SUBS R0, R0, #1
    BNE LOOP
```

0 ✗	0 ✓	0 ✗	0 ✗
1	5	for ever	0 (code is skipped)

Responses are hidden

CR

Menti

Lecture 4 - Branching

Choose a slide to present

Instructions

What is the value of register R0 after the following code is executed?

How many times does the loop LOOP run?

Branch Example
if-then-else

```
if (g==h) f=g-h;
else f=g+h;
```

```
cmp r2, r3 @ r2 == r3
bne else
sub r0, r2, r3
b if_end
else:
add r0, r2, r3
if_end:
```

<u>register mapping</u>

```
f: r0
g: r2
h: r3
```

Branch Example
if-then-else Alternative

```
if (g==h) f=g-h;
else f=g+h;
```

```
        cmp r2, r3      @ r2 == r3
        beq if
        add r0, r2, r3
        b if_end
if:
        sub r0, r2, r3
if_end:
```

register mapping

```
f: r0
g: r2
h: r3
```

Branch Example while

```
int pow = 1, x = 0;
while (pow != 128) {
    pow = pow * 2;
    x = x + 1;
}
```

```
    mov r0, #1
    mov r1, #0
WHILE:
    cmp r0, #128
    beq DONE
    lsl R0, R0, #1
    add R1, R1, #1
    b WHILE
DONE:
```

register mapping

pow:	r0
x:	r1

Conditional Instruction

- ARM assembly allows conditions on instruction.
  - ADDEQ, ADDNE
- Thumb-2 introduced the If-Then (IT) instruction.
  - providing conditional execution for up to four consecutive instructions.The conditions might all be identical, or some might be the inverse of the others.
  - Instructions within an IT block must also specify the condition code to be applied.

Conditional Execution Example

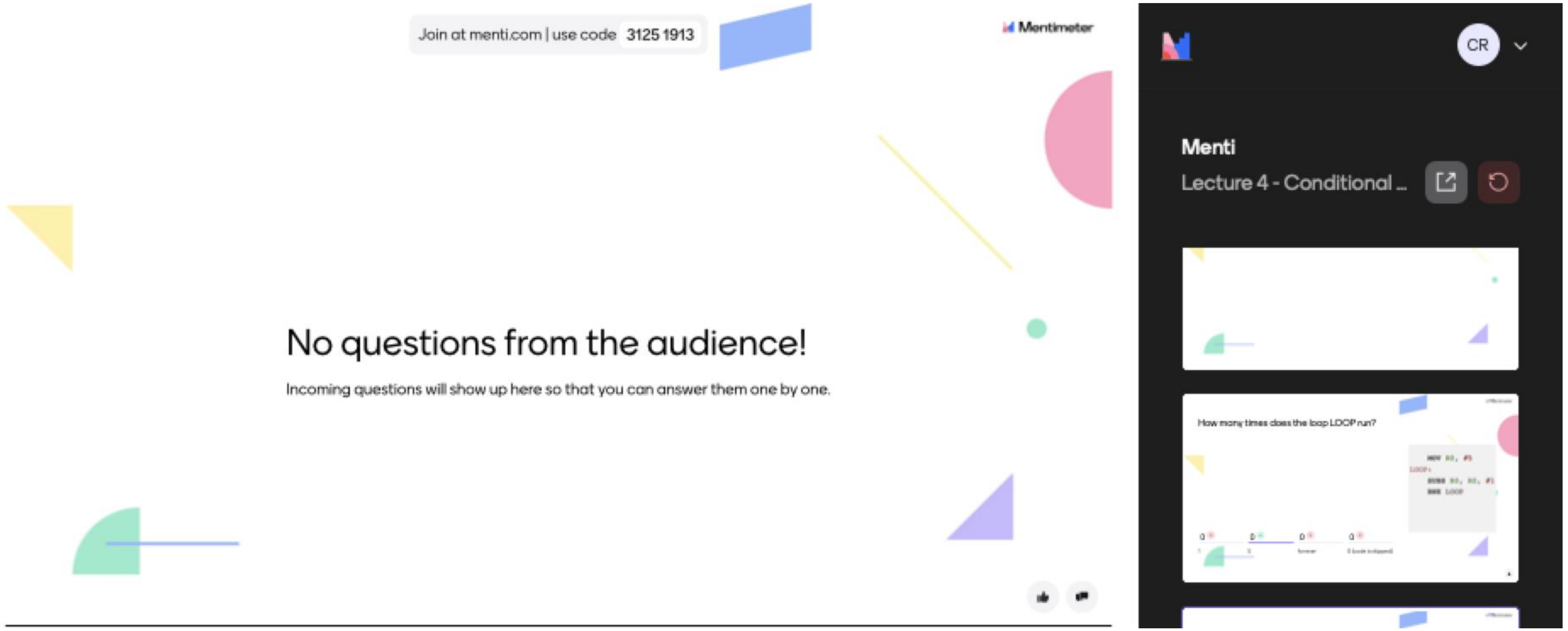
r2 = 0x80000000, r3 = 0x00000001

0x80000000 – 0x00000001 = 0x80000000 + 0xFFFFFFFF = 0x7FFFFFFF

C = 1	V=1	N=0	Z=0

```
cmp r2, r3
it eq
addeq r4, r5, #78 @ Z != 0
it hs
andhs r7, r8, r9 @ C == 1
it mi
orrmi r10, r11, r12 @ N != 0
ite lt
eorlt r12, r7, r10 @ C == 1 && Z != 0
eorge r12, r7, r10 @ C == 1 && Z != 0
```

Questions?


Join at menti.com | use code 3125 1913

No questions from the audience!

Incoming questions will show up here so that you can answer them one by one.

Mentimeter
CR
Menti
Lecture 4 - Conditional ...
How many times does the loop LOOP run?

Revision Question

- What is the value of R0, if R1 is 0 and if R1 is 1?

```
      CMP R1, #0
      BEQ SKIP
      ADD R0, R1, #5
SKIP:
```

- Which condition flag does BLE (Branch if Less or Equal) check?

- What is the value of R0, after executing the following code:

```
      MOV R0, #0
      MOV R1, #5
LOOP: ADD R0, R0, R1
      SUBS R1, R1, #1
      BNE LOOP
```

Conclusion

- Branching
  - PC manipulation
  - labels
- Conditional Branching
  - CPSR
  - Flags
- Conditional Execution
  - IT block
- Next
  - First program