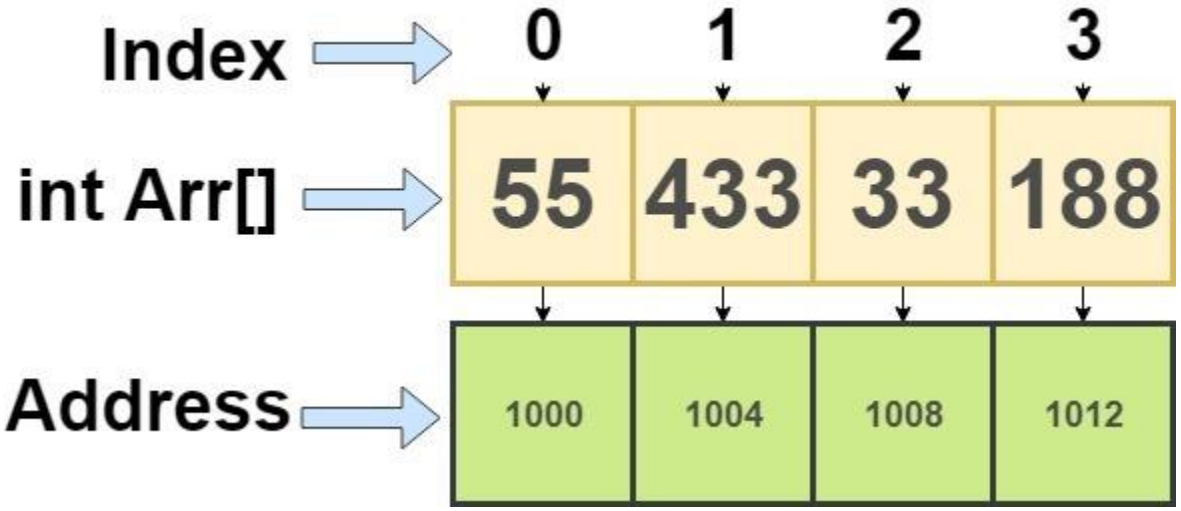
# RISC-V
# Conditional Operations for Implementing Loops (For/while/DoWhile)

Sparsh Mittal

# Insights

- Same assembly instructions (beq, bne, etc.) are used for implementing both
  - if conditions and
  - loop iterations (do, while, for)

# Finding address of variable




Base address of Arr is 1000
Arr[2] starts from 1008

Question: Let the base address of an array Arr be BaseAddress. The size of each variable is VariableSize bytes, find the address of the element Arr[i]
Answer: BaseAddress+VariableSize*i

# Solved Question: Convert "for" loop code

- Variables i in x28, j in x29. Base address of "long int" array A in x10. sum in x11.

```
for(int i=0; i<j; i++)
  sum = sum+A[i];
```

**Let us break it into simple operations in C/C++**

**C/C++ code**

```
i=0
LOOP: if (i>=j) goto EXIT
    offset = i*8
    addr = A+offset
    val = Memory[addr]
    sum = sum+val
    i= i+1
    goto LOOP
EXIT:
```

**RISC-V code**

```
addi x28, x0, 0
LOOP: bge x28, x29, EXIT
    slli x12, x28, 3
    add x12, x12, x10
    ld x7, 0(x12)
    add x11, x11, x7
    addi x28, x28, 1
    beq x0, x0, LOOP
EXIT:
```

# Solved Question: Convert “for” loop code

**Correct solution**

```
 addi x28, x0, 0
LOOP: bge x28, x29, EXIT
    slli x12, x28, 3
    add x12, x12, x10
    ld x7, 0(x12)
    add x11, x11, x7
    addi x28, x28, 1
    beq x0, x0, LOOP
EXIT:
```

**Incorrect solution**

```
add x28,x0,x0
Loop: slli x30,x28,3
add x30,x30,x10
ld x31,0(x30)
add x11,x11,x31
addi x28,x28,1
blt x28,x29,LOOP
beq x0,x0,EXIT
EXIT:
```

Here i<j is checked at the end of loop, which is incorrect

# What is difference between these two?

```
 addi x28, x0, 0
LOOP: bge x28, x29, EXIT
    slli x12, x28, 3
    add x12, x12, x10
    ld x7, 0(x12)
    add x11, x11, x7
    addi x28, x28, 1
    beq x0, x0, LOOP
EXIT:
```

```
 addi x28, x0, 0
EXIT: bge x28, x29, LOOP
    slli x12, x28, 3
    add x12, x12, x10
    ld x7, 0(x12)
    add x11, x11, x7
    addi x28, x28, 1
    beq x0, x0, EXIT
LOOP:
```

Answer: both are exactly same. The name of the label does not matter.

# Solved Question: Convert "while" loop

- i in x22, k in x24, Base address of "long int" array save in x25

```
while (save[i] ==k)
    i +=1
```

**Let us break it into atomic operations in C/C++**

**C/C++ code**

```
Loop: offset = i*8;
      addr = offset+save
      val = Memory[addr]
      if(val != k) goto Exit
      i = i+1
      goto Loop
Exit: …
```

**RISC-V code**

```
Loop: slli x10, x22, 3
      add  x10, x10, x25
      ld   x9, 0(x10)
      bne  x9, x24, Exit
        addi x22, x22, 1
      beq  x0, x0, Loop
Exit: …
```

# Solved Question: Convert "do while" loop

- i in x22, k in x24, Base address of "long int array" save in x25

```
do{
i +=1
 }while(save[i] ==k);
```

**Let us break it into atomic operations in C/C++**

**C/C++ code**

```
Loop: i = i+1
      offset = i*8;
      addr = offset+save
      val = Memory[addr]
      if(val != k) goto Exit
      goto Loop
Exit: …
```

**RISC-V code**

```
Loop: addi x22, x22, 1
      slli x10, x22, 3
      add  x10, x10, x25
      ld   x9, 0(x10)
      bne  x9, x24, Exit
      beq  x0, x0, Loop
Exit: …
```

# How will the previous code change on change in datatype?

- If the datatype is char/short/int/ “long int”, what changes come?
- In slli, we shifted by 3 for double, since double is 8B and 2^3 = 8
- For other datatypes, shift and load depending on their sizes.
- Rest of the code remains the same.

| RISC-V code | Short(2B) | Int/float(4B) | char(1B) |
|---|---|---|---|
| Loop: addi x22, x22, 1 | | | |
| slli x10, x22, 3 | slli x10, x22, 1 | slli x10, x22, 2 | mv x10 x22 #shift not reqd |
| add x10, x10, x25 | | | |
| ld x9, 0(x10) | lh x9, 0(x10) | lw x9, 0(x10) | lb x9, 0(x10) |
| bne x9, x24, Exit | | | |
| beq x0, x0, Loop | | | |
| Exit: … | | | |

# Convert C code to RISC-V

- while (array[i++] !=k) ;
- i be in x22, k in x24, base address of **short** datatype array in x25.

```
while:      slli x11, x22, 1       # calculate the offset address
            add x12, x25, x11 # base address + offset address
            lh x13, 0(x12) # load array[i]. lh because load half-word.
            addi x22, x22, 1     # i++
            beq x13, x24, exit # break the loop on equality
            beq x0, x0, while  # loop back to continue
exit:   NOP
```