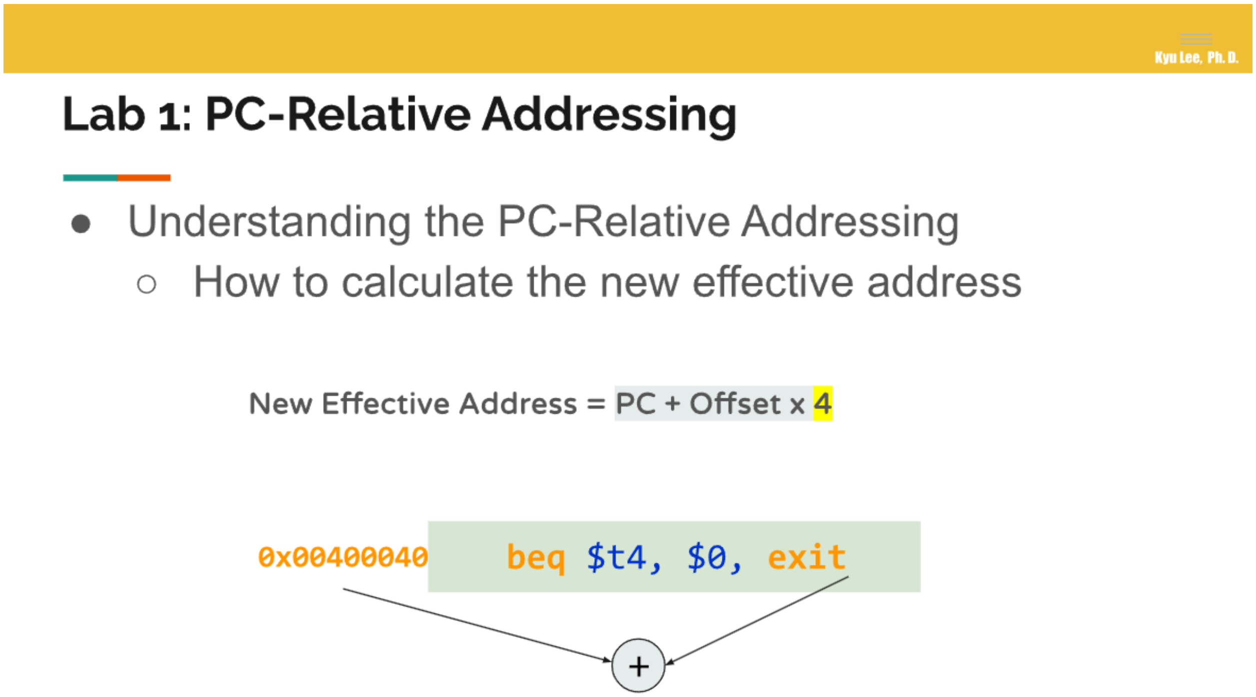
Assignment A P3 Lab 1 PC-Relative Addressing



Kyu Lee, Ph. D.

Lab 1: PC-Relative Addressing

- Understanding the PC-Relative Addressing
  - How to calculate the new effective address

New Effective Address = PC + Offset x 4

```
0x00400040    beq $t4, $0, exit
```

Elaboration:

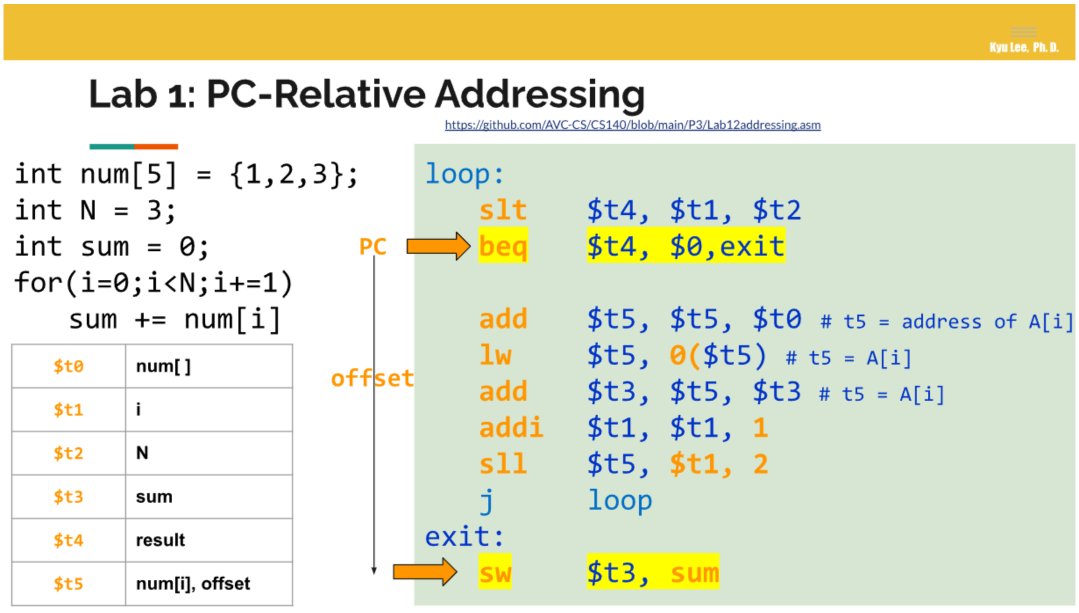
The effective address for branch instructions is calculated as

New Effective Address = Program Counter + Offset x 4

The example showcases this line of instruction

Beq $t4, $0, exit # Execute Exit branch if $t4 and $0 are equal

If the branch is taken, the program will use the calculation highlighted above, to jump. If the PC is 0x00400040, the target address of the branch is determined by adding offset x 4 to the current PC.



Kyu Lee, Ph. D.

Lab 1: PC-Relative Addressing

https://github.com/AVC-CS/CS140/blob/main/P3/Lab12addressing.asm

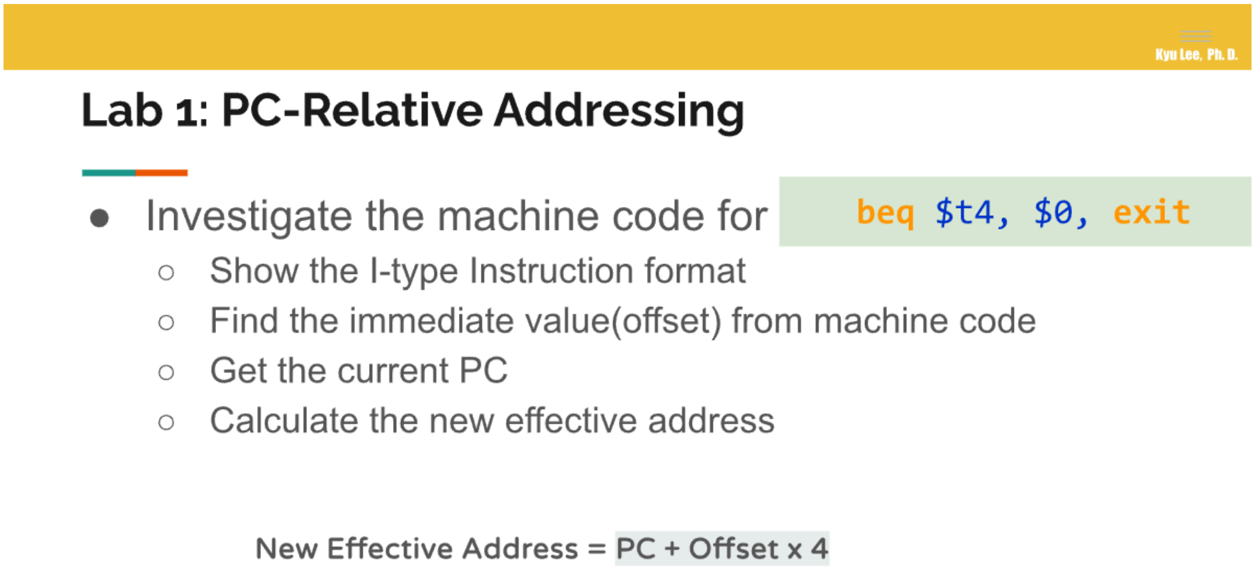
```
int num[5] = {1,2,3};
int N = 3;
int sum = 0;
for(i=0;i<N;i+=1)
    sum += num[i]
```

$t0	num[ ]
$t1	i
$t2	N
$t3	sum
$t4	result
$t5	num[i], offset

```
loop:
    slt     $t4, $t1, $t2
    beq     $t4, $0,exit

    add     $t5, $t5, $t0  # t5 = address of A[i]
    lw      $t5, 0($t5)  # t5 = A[i]
    add     $t3, $t5, $t3  # t5 = A[i]
    addi    $t1, $t1, 1
    sll     $t5, $t1, 2
    j       loop
exit:
    sw      $t3, sum
```

Elaboration:
PC-relative addressing allows branch instructions like beq to calculate the target address dynamically using the formula highlighted. The Offset is stored in the branch instructions and specifies how many instructions forward/backward to jump. For example, if an instruction is at 0x00400040 and the offset is +3, the target address is 0x00400040 + (3 x 4) = 0x0040004C. The program logic showcased on this slide is a loop that mimics a while structure using branch instructions (beq) and comparison (slt), iterating over the array to compute the sum. The calculation uses efficient address arithmetic with shifts and additions to index the array. Registers ($t0–$t5) are used to store intermediate values like loop counters, offsets, and data values, minimizing memory accesses. Data is fetched from memory (num[i]) only when needed and results are written back at the end.



Kyu Lee, Ph. D.

Lab 1: PC-Relative Addressing

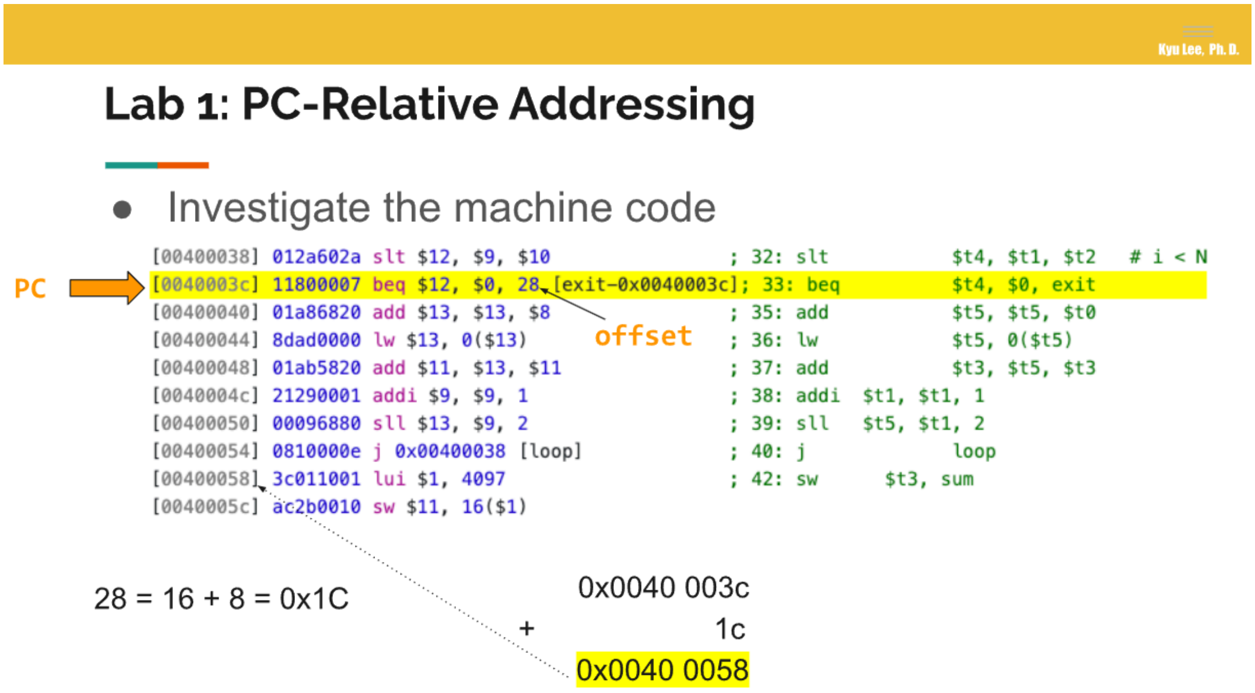
- Investigate the machine code for `beq $t4, $0, exit`
  - Show the I-type Instruction format
  - Find the immediate value(offset) from machine code
  - Get the current PC
  - Calculate the new effective address

New Effective Address = PC + Offset x 4

Elaboration:
This lab analyzes how PC-relative addressing works. The instruction provided
beq $t4, $0, exit   # branch off to exit if $t4 and $0 are equal

I-Type Instruction format
opcode | rs | rt | immediate
Opcode: beq has opcode 000100 in binary
rs: Source register, $t4 in this case, 01100 in binary
rt: Target register, $0 in this case, 00000 in binary
Immediate: 16-bit signed value (offset)
The offset is a relative value that determines how far to jump from the current program counter (PC). It is extracted from the machine code representation of the instruction. The offset is multiplied by 4 to get the byte address (since MIPS instructions are word-aligned).
If the instruction was at address 0x00400010, and exit is at 0x00400020.
Offset = (0x00400020 - 0x00400010 - 4) / 4
Offset = **4**   **.** 000100|01100|00000|0000 0000 0000 0100   Convert to hex.
Machine code is then, **0x11800004**

Let’s say 0x00400014 is the current PC.
Offset is 4. 0x00400014 + (4 x 4) = **0x00400024**
The program will jump to 0x00400024 if the condition ($t4 == $0) is true



Lab 1: PC-Relative Addressing

- Investigate the machine code

```
[00400038] 012a602a slt $12, $9, $10                  ; 32: slt           $t4, $t1, $t2   # i < N
[0040003c] 11800007 beq $12, $0, 28 [exit-0x0040003c]; 33: beq           $t4, $0, exit
[00400040] 01a86820 add $13, $13, $8                  ; 35: add           $t5, $t5, $t0
[00400044] 8dad0000 lw $13, 0($13)                    ; 36: lw            $t5, 0($t5)
[00400048] 01ab5820 add $11, $13, $11                 ; 37: add           $t3, $t5, $t3
[0040004c] 21290001 addi $9, $9, 1                    ; 38: addi  $t1, $t1, 1
[00400050] 00096880 sll $13, $9, 2                    ; 39: sll   $t5, $t1, 2
[00400054] 0810000e j 0x00400038 [loop]               ; 40: j             loop
[00400058] 3c011001 lui $1, 4097                      ; 42: sw      $t3, sum
[0040005c] ac2b0010 sw $11, 16($1)
```

28 = 16 + 8 = 0x1C

```
  0x0040 003c
+          1c
  0x0040 0058
```

The instruction beq $t4, $0, exit at address 0x0040003c uses PC-relative addressing to calculate the branch target. The instruction's immediate field (7) represents an offset in words, which is multiplied by 4 to get a byte offset of 28 (0x1C in hexadecimal). The branch target address is computed as the address of the next instruction (PC + 4 = 0x00400040) plus the byte offset, resulting in 0x00400058. If the condition $t4 == $0 is true, the program will branch to this calculated address.