CASPER

CS230: Digital Logic Design and Computer Architecture

Lecture 6: MIPS instructions contd.

https://www.cse.iitb.ac.in/~biswa/courses/CS230/main.html

Phones (smart/non-smart) on silence plz, Thanks

Logistics

When: Quiz-I, January 27, 11:15 AM

Where: LA-001 and CC-105

Report by 11.10 AM

Duration: 1 hour

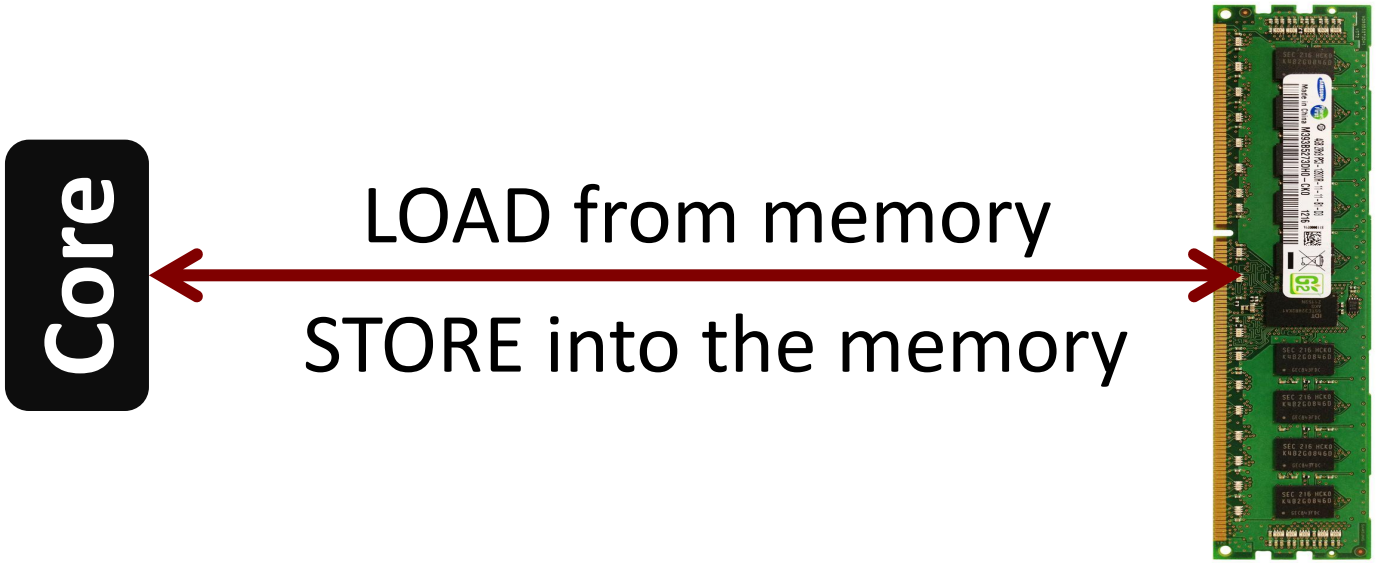
Make sure you clear all your doubts on

Monday 1:30 to 5:30 PM

Recap

- ISA
- Assembly
- Machine level
- Instructions
-

Memory Instructions




```
lw $t0, 1($a0)      # $t0 = Memory[$a0 + 1]
sw $t0, 1($a0)      # Memory[$a0 + 1] = $t0
```

Stored Program & Von Neumann

Memory

4GB of Memory (DRAM)

Say, a word: four bytes

How to access instructions: Program Counter (PC)

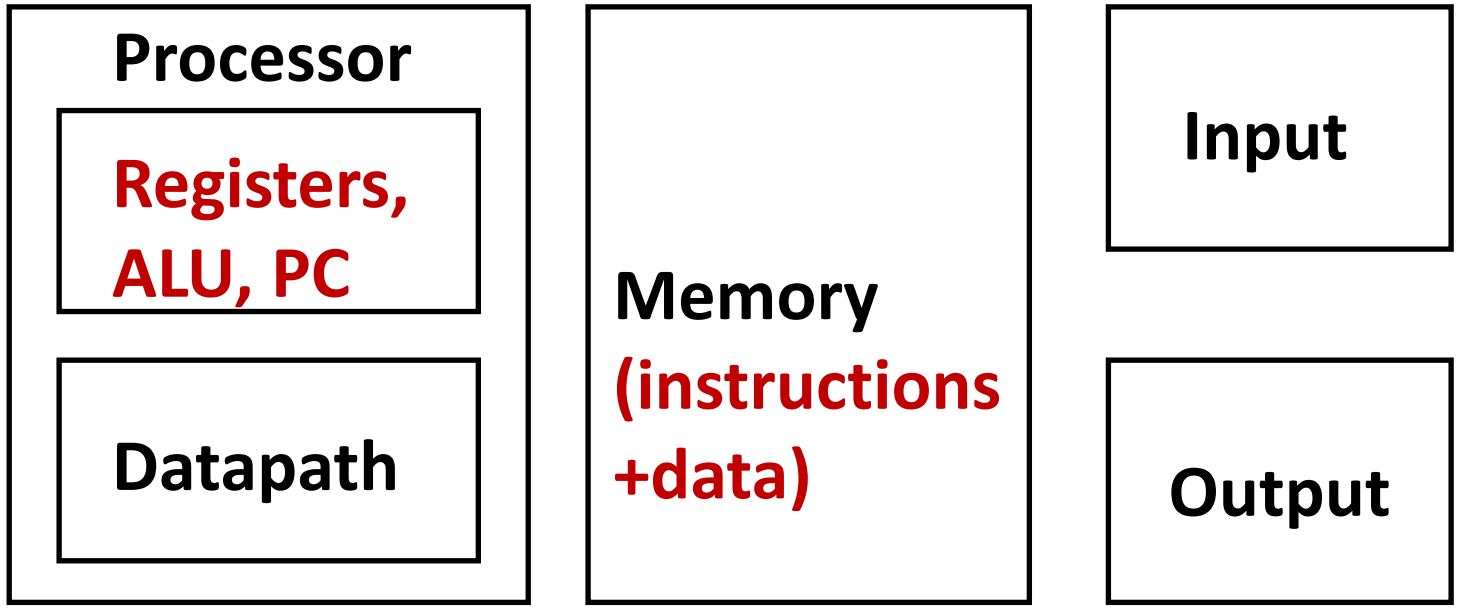
A register that stores the address of the instruction

32-bit processor: addresses are of width 32 bits (devil is in the details ☺)

So the processor fetches PC, PC+4, PC+8, ….. in a sequential order

1946 onwards

Since 1946 all computers have had 5 components



Example (Remember PC for the time being)

PCX: lw

PCY: add

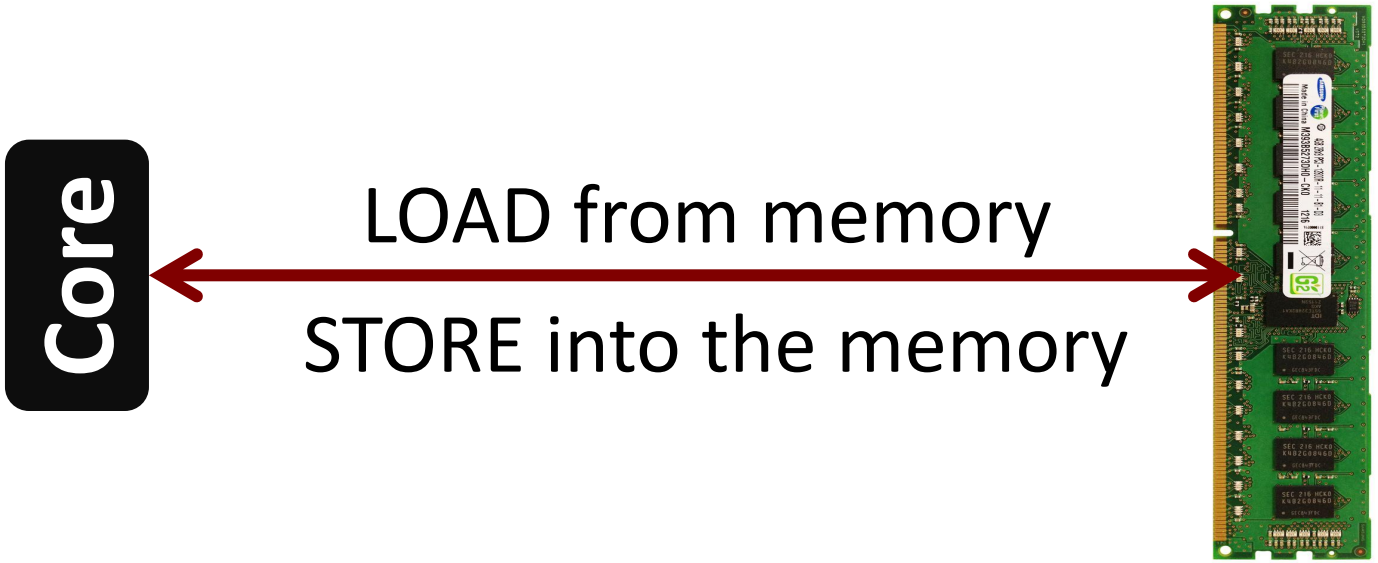
PCZ: lui

PCZ=PCY+4 and PCY = PCX+4

Why Memory? Why Not Registers?

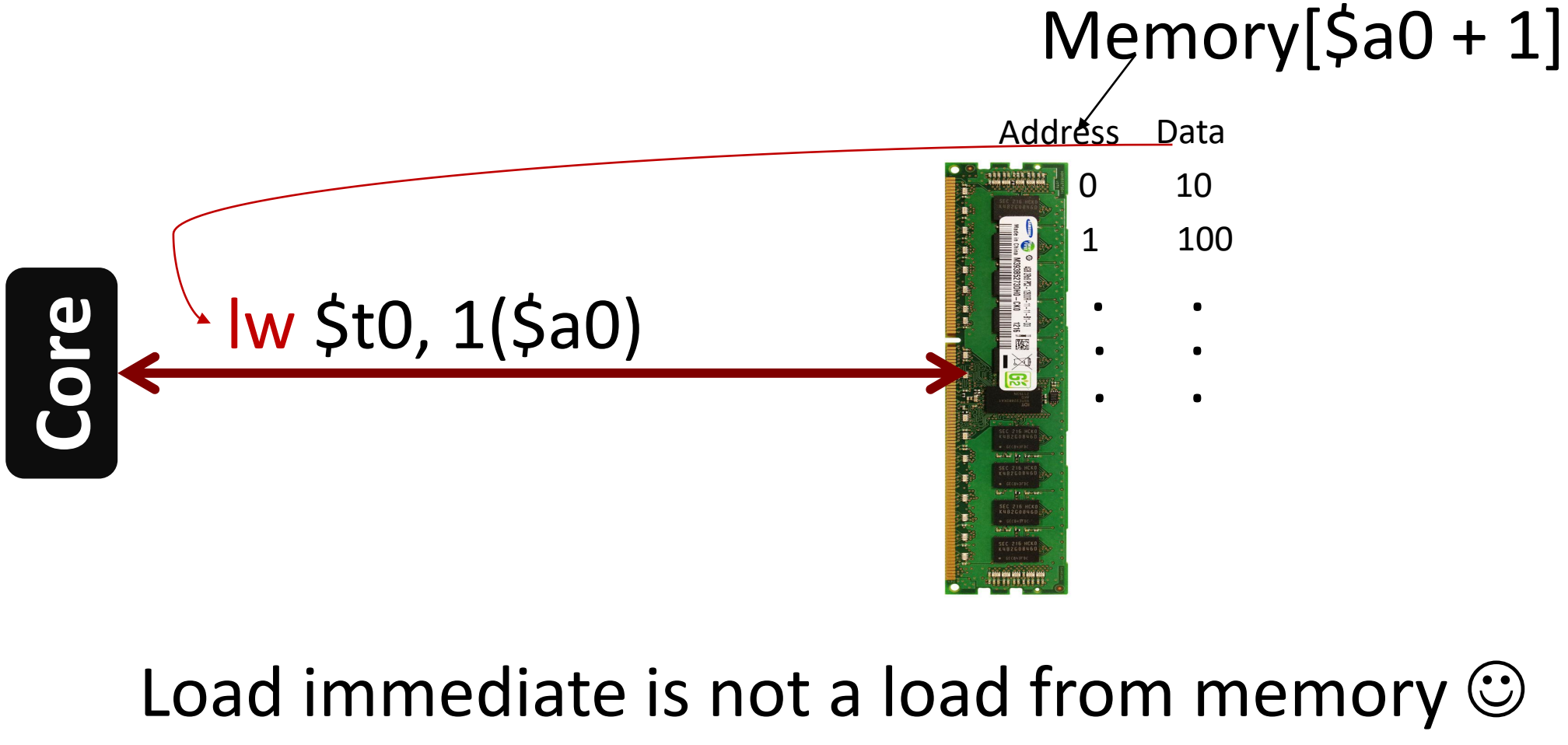
- Registers are limited. More #registers, higher access time.
- How? we will see sooner than later.
- Let’s focus on the data part now. How to access data for our instructions?

Memory Instructions



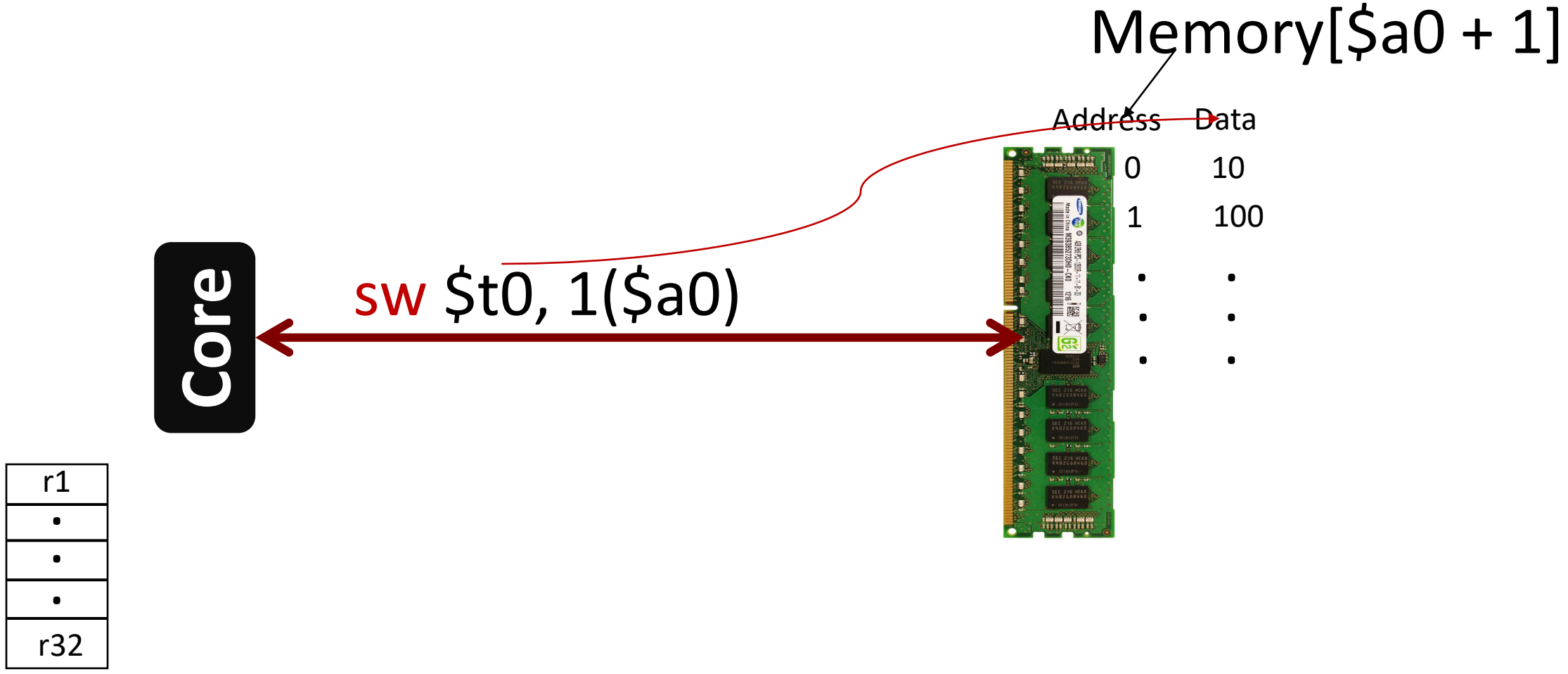

```
lw $t0, 1($a0)      # $t0 = Memory[$a0 + 1]
sw $t0, 1($a0)      # Memory[$a0 + 1] = $t0
```

LOAD From the Memory (data-transfer insts)



Load immediate is not a load from memory ☺

STORE



Both instructions and data from memory

g = h + A [8];

PCX: lw $t0, 8($3) # A[8]

PCY: add $s1, $s2, $t0 # g = h + t0

PCY = PCX+4

A quick recap

Von Neumann (stored program) concept

As registers are limited, data can be there in the registers or in the memory

Register accesses are through register names/numbers

Memory accesses are through addresses stored in registers

Let's move on: Decision Making Instructions

- Decisions: if, else ….

Two instructions:

beq (branch equals to) and

bne (branch not equals to)

```
beq $t0, $t1, L1
bne $t0, $t1, L1
```

Branch Instructions: Conditional branches

beq $t0, $t1, L1

goto L1 (statements labeled as L1) if $t0 equals $t1

bne $t0, $t1, L1

goto L1 (statements labeled as L1) if $t0 does not equal to $t1

The slt instruction (Set on less than)

if (a < b) // beq and bne won't work here

  c=1

else

  c=0

slt $t3, $t1, $t2 // t1 and t2 contain a and b

We can slti too; one of the operand will be a constant

Loops: How to deal with it?

```
while(CS230[i] == k)
    i+=1;
```

say i and k are in $s3 and $s5, and the

base of CS230 in $s6

Loops continued

```
while(CS230[i] == k)
    i+=1;
```

1. LOAD CS230[i], base address of CS230 is in $s6
2. We need to go to CS230[i]
3. Assuming CS230 is an integer array, each index is of 4 bytes. We need to go to CS230 [i*4 bytes]

Loops contd. ($s3=i, $s5=k, $s6=base address)

```
sll $t1, $s3, 2          // i*4
add $t1, $t1, $s6   // address of CS230[i]
lw $t0, 0($t1)        // t0 = CS230[i]
bne $t0, $s5, Exit // go to Exit if CS230[i] not equals to k
addi $s3, $s3,1     // i=i+1

Exit:                        // do nothing
```

```
while(CS230[i] == k)
  i+=1;
```

Where is the Loop?

Loops continued

```
Loop: sll $t1, $s3, 2        // i*4
      add $t1, $t1, $s6   // address of CS230[i]
      lw $t0, 0($t1)       // t0 = CS230[i]
      bne $t0, $s5, Exit // go to Exit if CS230[i] not equals to k
      addi $s3, $s3,1     // i=i+1

Exit:                      // do nothing
```

```
while(CS230[i] == k)
  i+=1;
```

How to jump to the Loop?

Loops continued

```
Loop: sll $t1, $s3, 2          // i*4                    while(CS230[i] == k)
      add $t1, $t1, $s6   // address of CS230[i]          i+=1;
      lw $t0, 0($t1)        // t0 = CS230[i]
      bne $t0, $s5, Exit // go to Exit if CS230[i] not equals to k
      addi $s3, $s3,1     // i=i+1
      j      Loop             // go to loop. j here is jump


Exit:                    // do nothing
```

Textbook Chapter 2 P&H

Coffee Credits

Lisan 210050076

haben Sie einen guten Tag