MIPS Instruction Set Architecture

Dr. Ahmed H. Zahran

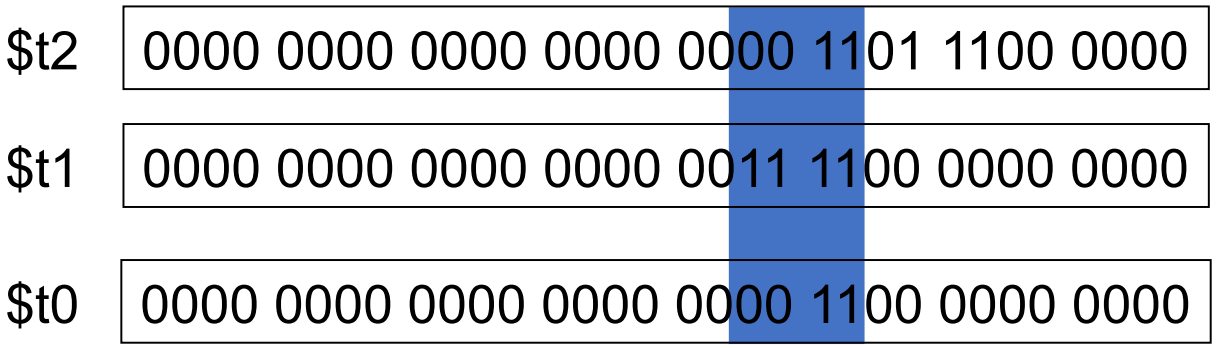
WGB 182

a.zahran@cs.ucc.ie

AND Operations

- Useful to **mask** bits in a word
  - Select some bits, clear others to 0

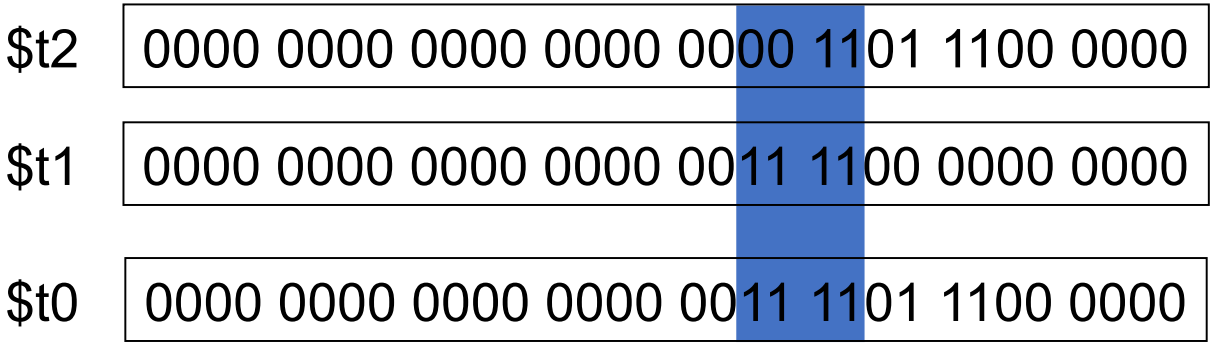
AND $t0, $t1, $t2



OR Operations

- Useful to **set** some bits to 1 in a word and leave others unchanged

  OR $t0, $t1, $t2



NOT Operations

- Useful to **invert** bits in a word
  - Change 0 to 1, and 1 to 0
- MIPS has NOR 3-operand instruction
  - a NOR b == ***NOT*** (a OR b)

NOR $t0, $t1, $zero

Register 0: always read as zero

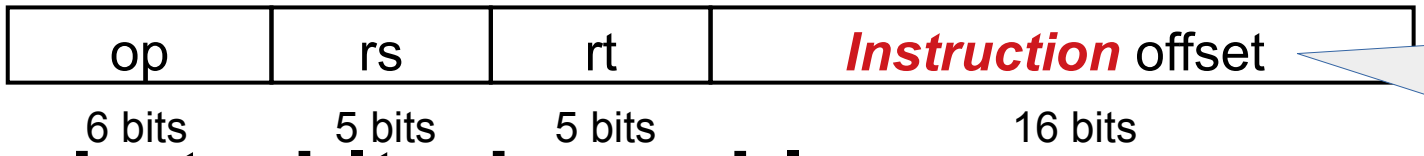
$t1	0000 0000 0000 0000 0011 1100 0000 0000
$t0	1111 1111 1111 1111 1100 0011 1111 1111

Objectives

- Explore branching instructions
- Understand procedure execution at processor level
- Understand how MIPS accommodate large constants

Branch Operations

- Branch to a labelled instruction: ***conditional and unconditional***
- ***Conditional branch*** instructions:
  1. ***beq*** rs, rt, L1 # if (rs == rt) branch to instruction labeled L1;
  2. ***bne*** rs, rt, L1 # if (rs != rt) branch to instruction labeled L1;



Target address = PC + offset × 4 [Relative address]

3. **bgtz, bltz, bgez, blez**

Why not to use BLT?

too complicated to implement → would stretch the clock cycle time or would take extra clock cycles per instruction

Solution: use two basic instructions to execute its logic :)

More Conditional Operations

- Set result to 1 if a condition is true
  - Otherwise, set to 0
- ***slt*** rd, rs, rt # if (rs < rt) rd = 1; else rd = 0;
- ***slti*** rt, rs, const #if (rs < constant) rt = 1; else rt = 0;
- Use in combination with beq, bne

```
slt $t0, $s1, $s2  # if ($s1 < $s2)
bne $t0, $zero, L  #   branch to L if $S1<$S2
```

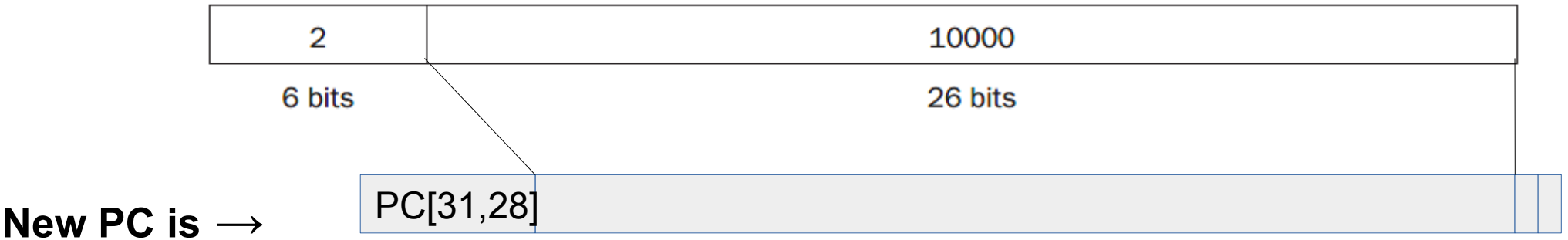
Signed vs. Unsigned

- Signed comparison: ***slt***, ***slti***
- Unsigned comparison: ***sltu***, ***sltui***
- Example
  - $s0 = 1111 1111 1111 1111 1111 1111 1111 1111
  - $s1 = 0000 0000 0000 0000 0000 0000 0000 0001
  - ***slt*** $t0, $s0, $s1 # signed
    - –1 < +1 ⇒ $t0 = 1
  - ***sltu*** $t0, $s0, $s1 # unsigned
    - +4,294,967,295 > +1 ⇒ $t0 = 0

Branch Operations

- Branch to a labelled instruction: ***conditional and unconditional***
- ***Unconditional*** branch instruction
  - ***j*** L1 # jump to instruction labelled L1

J-format Instruction



If Statement Example

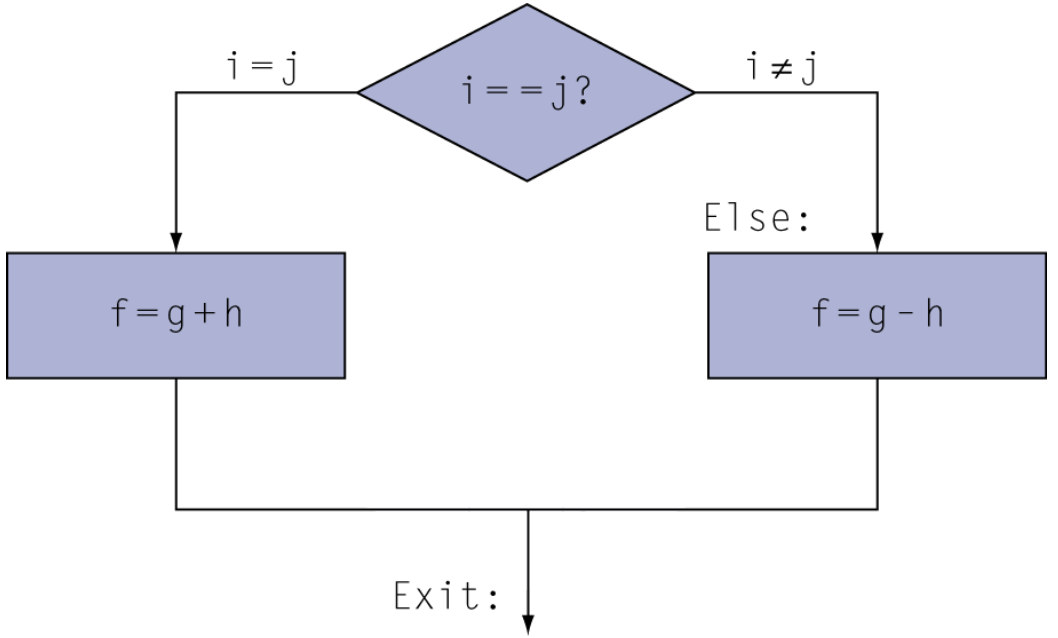
- C code:

  if (i==j) f = g+h;
  else f = g-h;

  - **f, g, … in $s0, $s1, …**

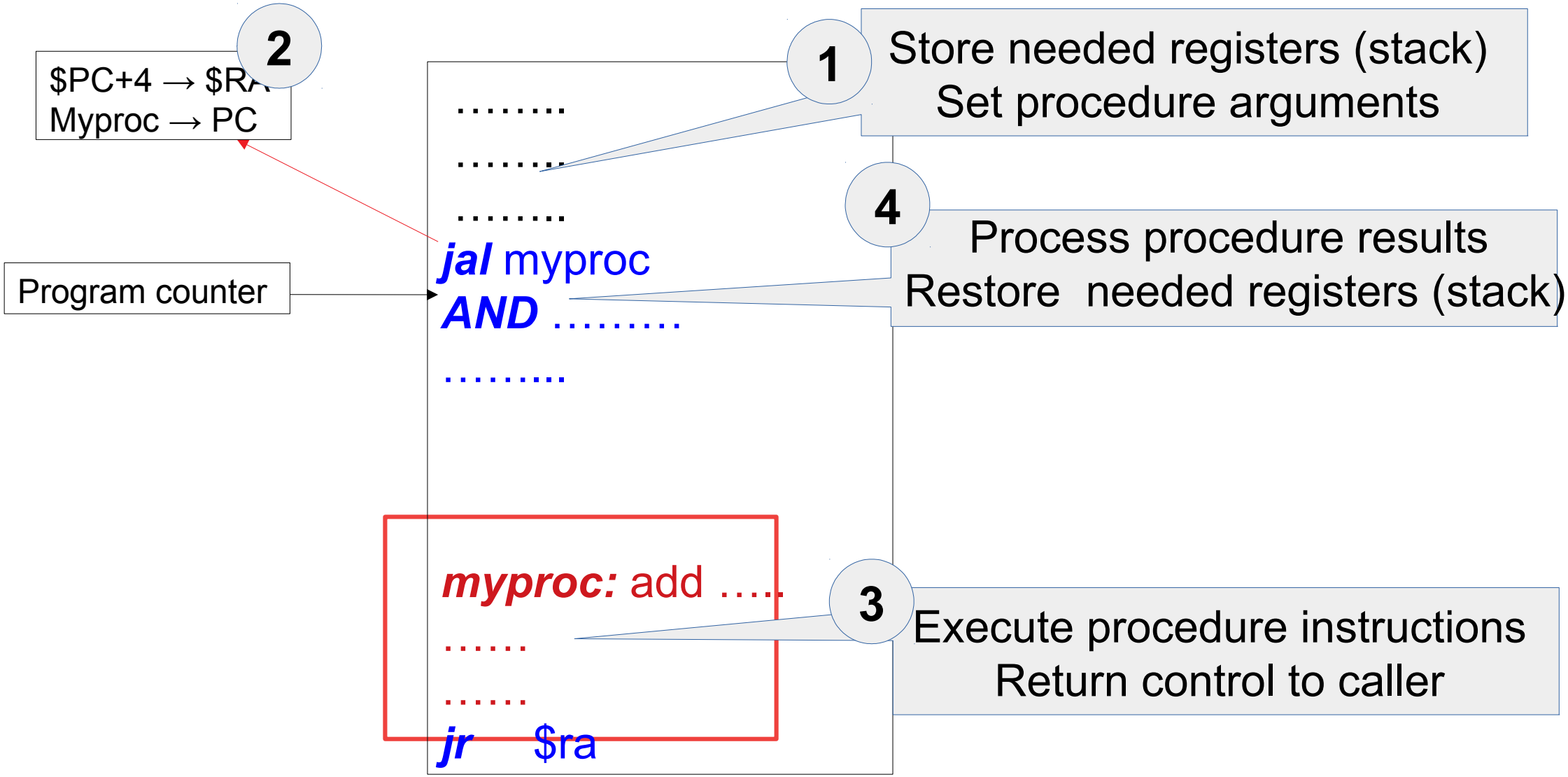
- Compiled MIPS code:

```
      bne $s3, $s4, Else
      add $s0, $s1, $s2
      j   Exit
Else: sub $s0, $s1, $s2
Exit:
```



Assembler calculates addresses

Procedure Calling



Procedure Calling

- Procedure allows programmers to concentrate on a portion of a task at a time
  - Includes parameter to interface to the rest of the program
- Steps applied for calling a procedure
  1. Place arguments in registers
  2. Transfer control to procedure
  3. Acquire storage for procedure
  4. Perform procedure’s operations
  5. Place result in register for caller
  6. Return to place of call

MIPS Procedure Instructions

- ***procedure call***: jump and link (jal)

  jal ProcedureLabel
  - Address of ***following instruction*** is put in $ra
  - Jumps to target address
- ***Procedure return***: jump register (jr)

  jr $ra
  - Unconditional jumps
  - Copies $ra to program counter
  - Can also be used for computed jumps

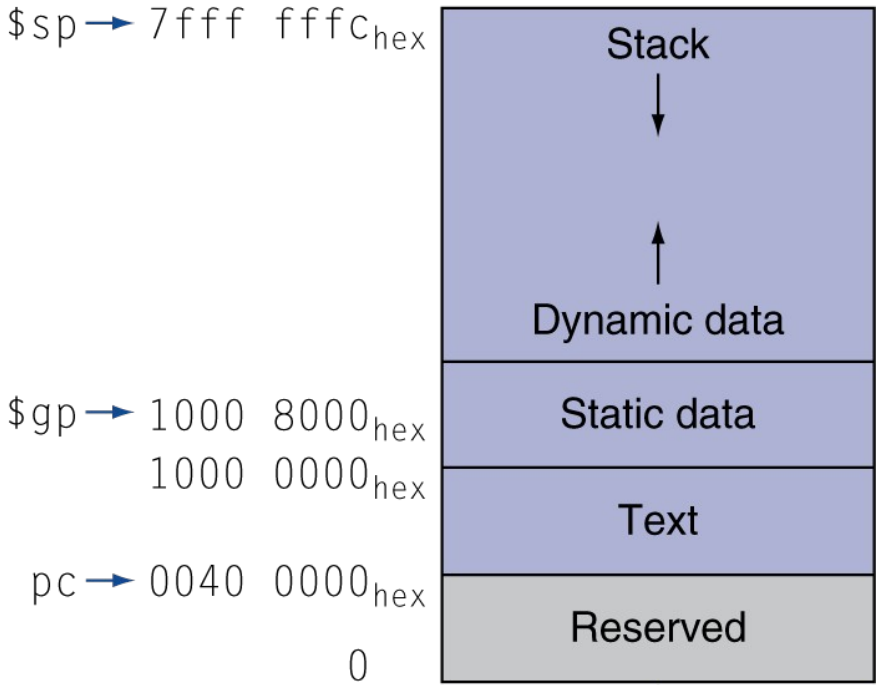
MIPS Register Map for Procedures

- $a0 – $a3: ***arguments registers*** for passing parameters (reg's 4 – 7)

- $v0, $v1: registers for **result values** (reg's 2 and 3)

- **Using more registers**
  - $t0 – $t9: temporaries [Can be overwritten by callee]
  - $s0 – $s7: saved registers [Must be saved/restored by callee]

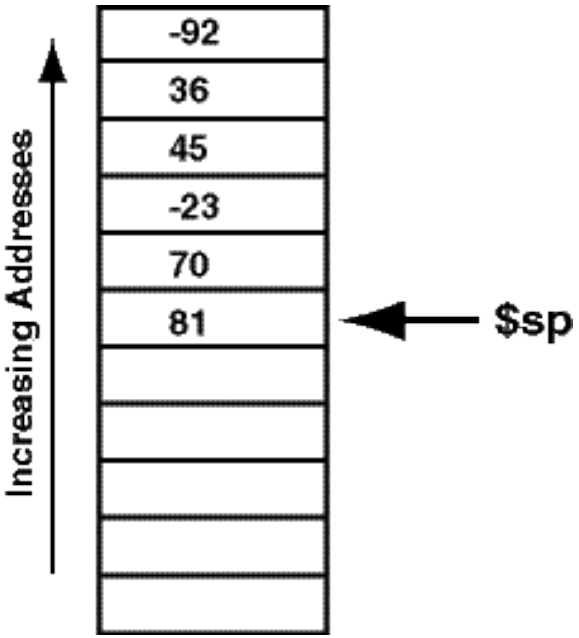
Memory Layout



- Text: program code
- Static data: global variables
  - e.g., static variables in C, constant arrays and strings
  - $gp (global pointer register)
- Dynamic data: heap
  - E.g., malloc in C, new in Java

Stack

- A last-in-first-out queue for storing register content
  - Stack pointer ($SP) points to the most recent allocated address in stack
  - MIPS stack is managed ***manually***



Reading

- Sections 2.8 – 2.9