CC311 Computer Architecture

Lecture 8

MIPS

Review Example

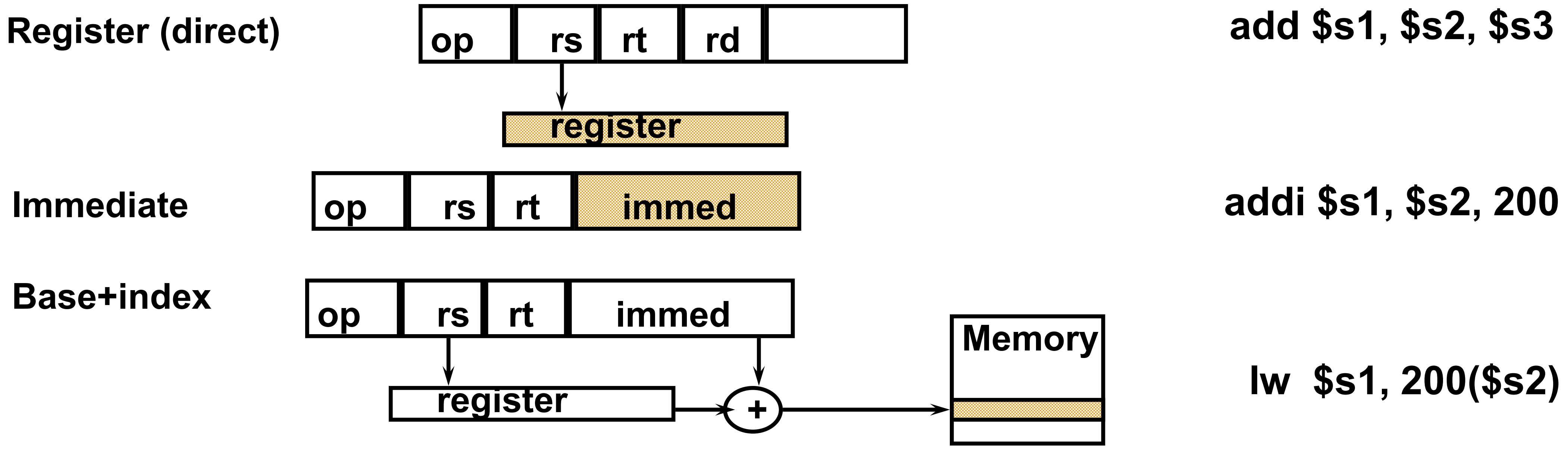
```
#z[i+4]=(a*b)- (e[i+2] *g);

   Consider a, z in memory as byte, e in memory as half
```

MIPS Instruction Types

- Arithmetic/Logical
  - R-type: destination and two source registers, shift amount
  - I-type: 16-bit immediate with sign/zero extension
- Memory Access
  - load/store between registers and memory
  - word, half-word and byte operations
- Control flow
  - conditional branches: pc-relative addresses
  - jumps: fixed offsets, register absolute

MIPS Addressing Modes

Addressing modes specify the location of data used by an instruction. Data can be in registers, memory or immediate (within the instruction itself).



Branch Classification

There are two basic types of branches:

- **Unconditional**: *Always* go to the specified address without any condition
- **Conditional**: go to the specified address if defined condition is true; otherwise, execute the next instruction

target addresses can be specified in the same way as other operands (combination of registers, immediate constants, or memory locations), based on what is supported in the ISA.

Jump Instructions

- Unconditional branch
- Two types of instructions:
  - R-type
    - JR $31
    - JALR $3
  - J-type
    - J address

Jump Instructions

- The default form of the jump instruction is: j label
- "jump" always goes to a labeled memory address.
- The next instruction executed at memory location "label."

  - This transfer is unconditional.
- Examples:
  - j end – The next instruction executed is the one labeled "end"
  - j go – The next instruction to be executed is labeled "go."
- There is NO option on jump instructions. Again, a jump is always to go a labeled location.
- Jump and branch instructions are the reason why instructions are labeled in the program.

Branch Classification

There are two basic types of branches:

- **Unconditional**: *Always* go to the specified address without any condition
- **Conditional**: go to the specified address if defined condition is true; otherwise, execute the next instruction

target addresses can be specified in the same way as other operands (combination of registers, immediate constants, or memory locations), based on what is supported in the ISA.

Branch Instructions

- **Branch instructions (usually used with jump instructions) enable MIPS programmers to incorporate decision-making capability into a program.**
- **For that, they known as "program control" instructions as they support the capability of a program to choose when we need to change operation.**
- **In general, a branch makes a comparison. If the result of the comparison is true, the next instruction executed is at another memory location not the next location.**
- **If the result of the comparison is false, the program executes the next instruction following the branch.**

Conditional branching

```
#Basicinstructions
beq    $t1,$t2, label          #if($t1==$t2)goto label
bne    $t1,$t2, label          #if($t1!=$t2)goto label

bgez   $t1, label              #if($t1>=0)goto label
bgtz   $t1, label              #if($t1>0)goto label
blez   $t1, label              #if($t1<=0)goto label
bltz   $t1, label              #if($t1<0)goto  label

#Macroinstructions
beqz   $t1, label              #if($t1==0)goto label
bnez   $t1, label              #if($t1!=0)goto label

beq    $t1, 123, label         #if($t1==123)goto label
bne    $t1, 123, label         #if($t1!=123)goto label

bge    $t1,$t2, label          #if($t1>=$t2)goto label
bgt    $t1,$t2, label          #if($t1>$t2)goto label
bge    $t1, 123, label         #if($t1>=123)goto label
bgt    $t1, 123, label          #if($t1>123)goto label
```

and similarly for `ble` and `blt`

Reading strings into memory

Jumps and conditional branches

Branching control structures

- If-then-else and if-then statements

Looping control structures

- Do-while, while, and for loops
- Break and continue, indefinite loops

Arrays

- For-each loop
- Switch statement

Branching

High level language

```
if cond then
    ...
    ...
    ...
    ...
    ...
else
    ...
    ...
    ...
    ...
end if
```

Assembly language

translation of condition, terminating with the label of then block (`Thenlabel`)

```
         ...   }
         ...   }  Else block
         ...   }
         ...   }
       j endlabel
Thenlabel:
        ...    }
        ...    }  then block
        ...    }
        ...    }
Endlabel:
         ...   }
         ...   }  Rest of program
         ...   }
```

If-then-else statement

Example

```
#if(m < n+5)
#m++
#n  = n/m
#Registermappings:
#m:$t0,n:$t1
        Addi $t2,$t1, 5      #tmp = n+5
        blt  $t0,$t2, then   #if(m<tmp)
        j    end
  then: addi $t0,$t0, 1       #(then block)m++
  end:  div  $t1,$t0          # n/m
        mflo $t1
```

If-then-else statement

Example

```
#if(m < n+3)
# m = m+1
#else
#m = m+2
#n = n+m
#Registermappings:m:$t0,n:$t1
        addi $t2,$t1, 3     #tmp = n+3
        blt  $t0,$t2, then  #if(m < tmp)
        addi $t0,$t0, 2     #(elseblock)m = m+2
        j    end
  then: addi $t0,$t0, 1     #(thenblock)m = m+1
  end:  add  $t1,$t1,$t0    #n = n+m
```

If-then-else statement

Write a MIPS program to read the birth year of a candidate and determine whether he is eligible to cast his/her own vote.