Instruction Set

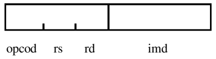
- Computation instructions
- Memory access instructions
- Control instructions
- System instructions

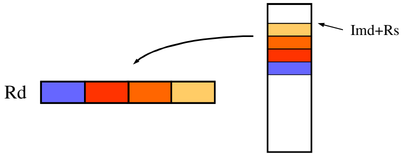
Instruction Set

- Loads
- Stores

Memory access

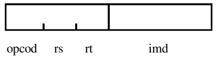
Assembly language	Action	Format
Lw Rd, Imd (Rs) Load word	4 bytes from memory at address *Imd+Rs* ⇨ Rd	I format

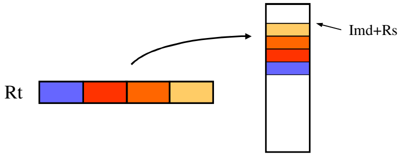




Memory access

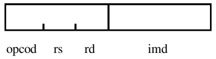
Assembly language	Action	Format
Sw Rt, Imd (Rs) Store word	4 bytes from Rt written into memory at address *Imd+Rs*	I format

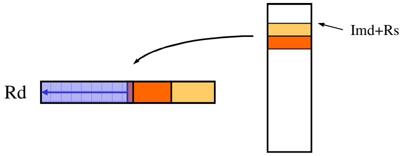




Memory access

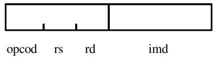
Assembly language	Action	Format
Lh Rd, Imd (Rs) Load half-word	2 bytes from memory at address *Imd+Rs* ⇨ Rd	I format

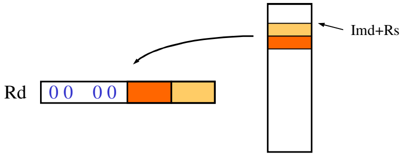




Memory access

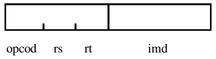
Assembly language	Action	Format
Lhu Rd, Imd (Rs) Load half-word unsigned	2 bytes from memory at address *Imd+Rs* ⇨ Rd	I format

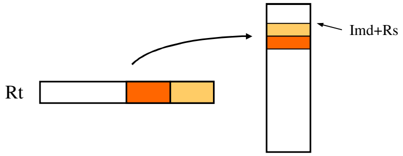




Memory access

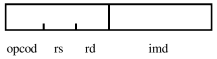
Assembly language	Action	Format
Sh Rt, Imd (Rs) Store half-word	2 bytes from Rt written into memory at address *Imd+Rs*	I format

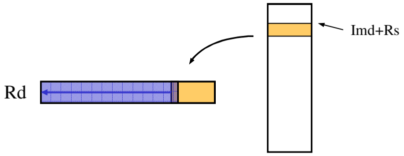




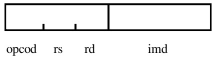
Memory access

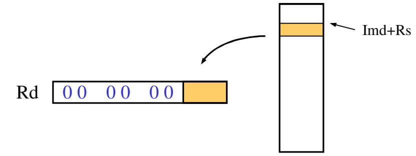
Assembly language	Action	Format
Lb Rd, Imd (Rs) Load byte	1 byte from memory at address *Imd+Rs* ⇨ Rd	I format



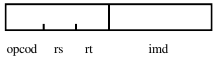


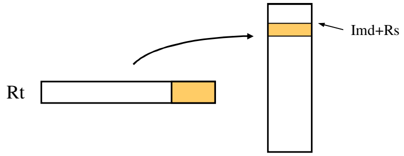
Memory access

Assembly language	Action	Format
Lbu Rd, Imd (Rs) Load byte unsigned	1 byte from memory at address *Imd+Rs* ⇨ Rd	 I format

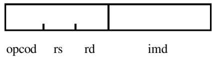


Memory access

Assembly language	Action	Format
Sb Rt, Imd (Rs) Store byte	1 byte from Rt written into memory at address *Imd+Rs*	 I format

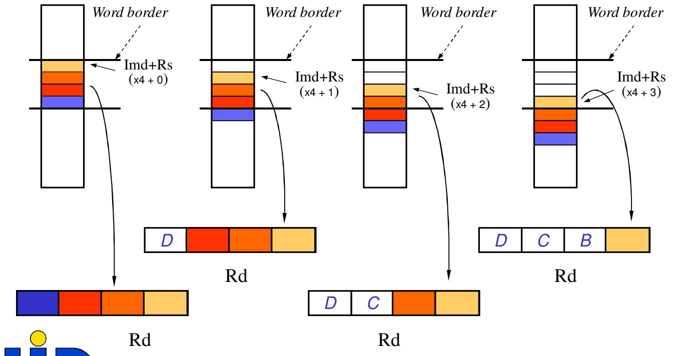


Memory access

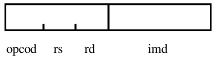
Assembly language	Action	Format
Lwr Rd, Imd (Rs) Load word right	1, 2, 3 or 4 bytes from memory at address *Imd+Rs* ⇨ Rd	 I format

Memory access

Load word right

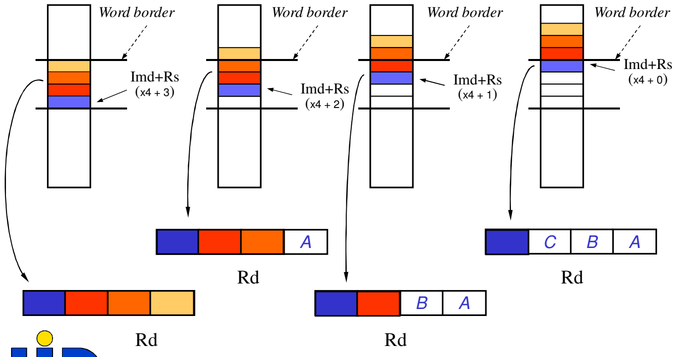


Memory access

Assembly language	Action	Format
Lwl Rd, Imd (Rs) Load word left	1, 2, 3 or 4 bytes from memory at address *Imd+Rs* ⇨ Rd	 I format

Memory access

Load word left



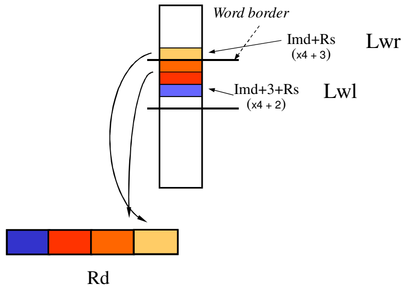
Control instructions

A couple of `Lwl`, `Lwr` (in any order) can be used

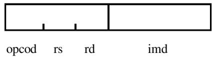
to load a word from any (unaligned) location

```
Lwl  Rd,  imd+3  (Rs)            Lwr  Rd,  imd    (Rs)
                          or
Lwr  Rd,  imd    (Rs)            Lwl  Rd,  imd+3  (Rs)
```

Memory access

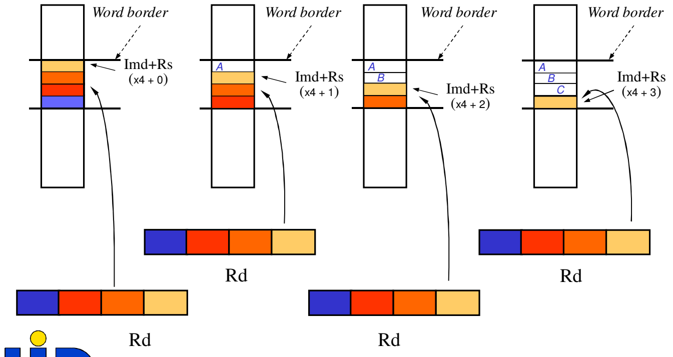


Memory access

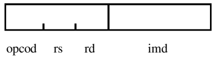
Assembly language	Action	Format
Swr Rt, Imd (Rs) Store word right	1, 2, 3 or 4 LsB of Rt written into memory at address *Imd+Rs*	 I format

Memory access

Store word right

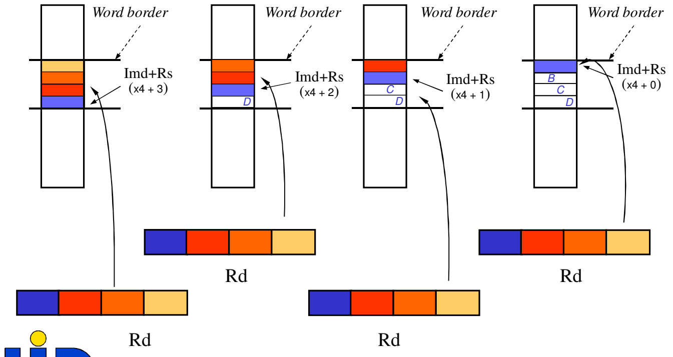


Memory access

Assembly language	Action	Format
Swl Rt, Imd (Rs) Store word left	1, 2, 3 or 4 MsB of Rt written into memory at address *Imd+Rs*	 I format

Memory access

Store word left



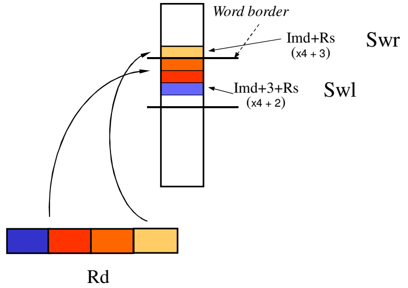
Control instructions

A couple of `Swl`, `Swr` (in any order) can be used

to load a word from any (unaligned) location

```
Swl  Rt,  imd+3  (Rs)          Swr  Rt,  imd    (Rs)
                          or
Swr  Rt,  imd    (Rs)          Swl  Rt,  imd+3  (Rs)
```

Memory access



Instruction Set

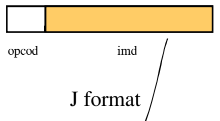
- Computation instructions
- Memory access instructions
- Control instructions
- System instructions

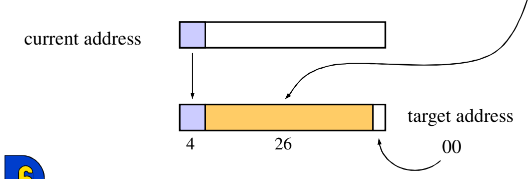
Control instructions

- Unconditional branches (jumps)
- Conditional branches

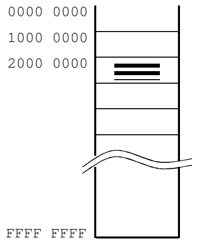
Control instructions

Assembly language	Action	Format
J label Jump	Go to the instruction labeled *label*	





Control instructions

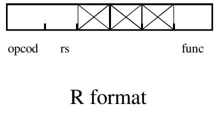


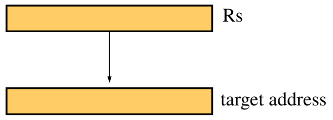
The 4 Msb of the address divide the memory space into 16 blocs

Each bloc = 256 Mbytes

Control instructions

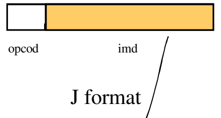
Assembly language	Action	Format
Jr Rs Jump register	jump to the address contained in Rs	

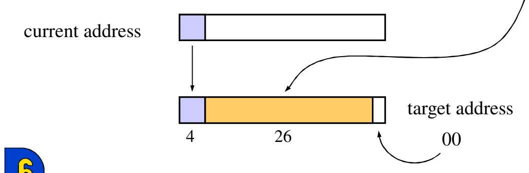


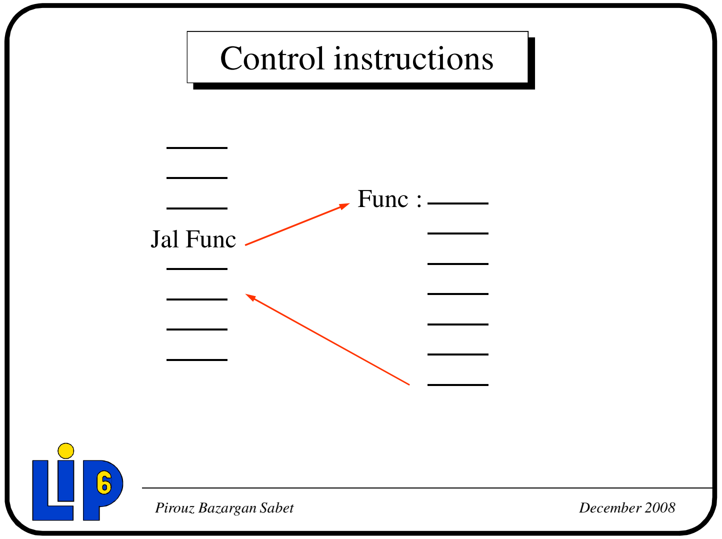


Control instructions

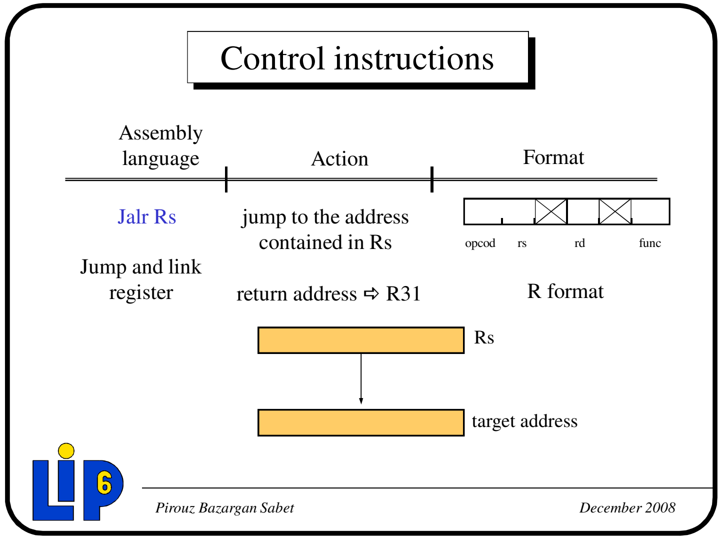
Assembly language	Action	Format
Jal label Jump and link	jump to the subroutine labeled *label* return address ➩ R31	





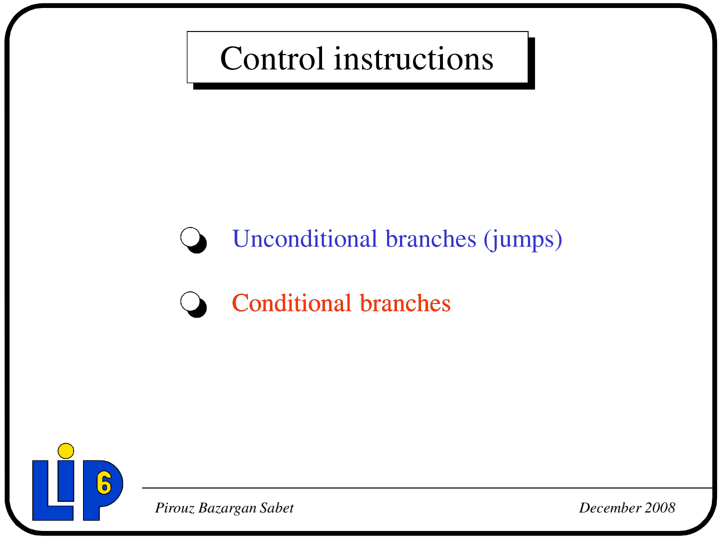


Control instructions



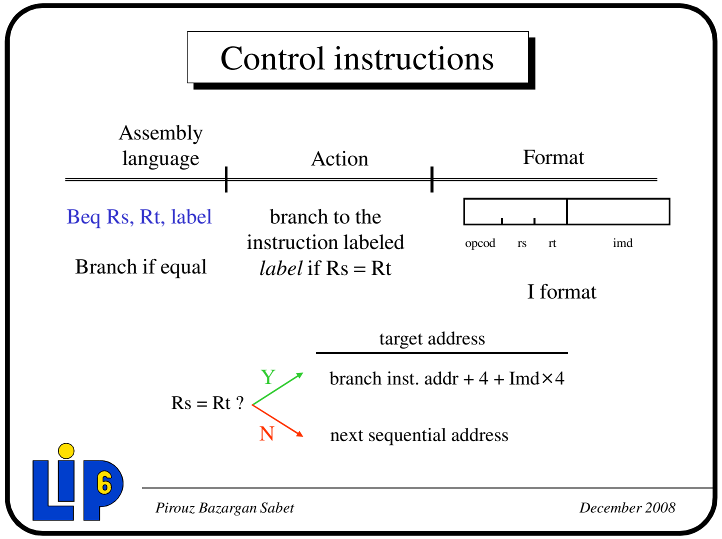
Control instructions

Assembly language	Action	Format
Jalr Rs Jump and link register	jump to the address contained in Rs return address ⇨ R31	opcod rs rd func R format



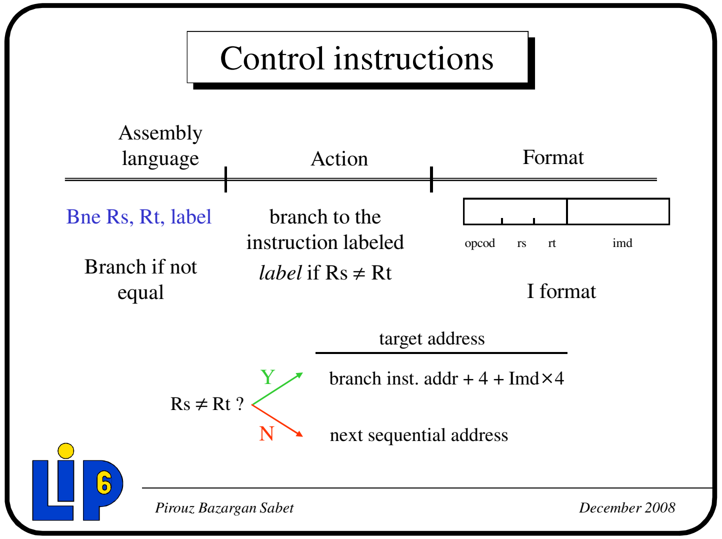
Control instructions

- Unconditional branches (jumps)
- Conditional branches



Control instructions

Assembly language	Action	Format
Beq Rs, Rt, label Branch if equal	branch to the instruction labeled *label* if Rs = Rt	opcod rs rt imd I format



Control instructions

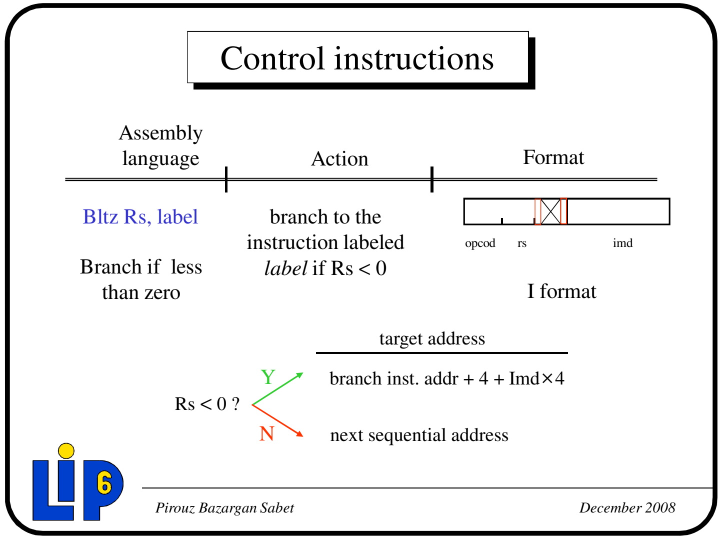
Assembly language	Action	Format
Bne Rs, Rt, label Branch if not equal	branch to the instruction labeled *label* if Rs ≠ Rt	opcod rs rt imd I format

target address

Rs ≠ Rt ? Y → branch inst. addr + 4 + Imd×4

N → next sequential address

Pirouz Bazargan Sabet *December 2008*



Control instructions

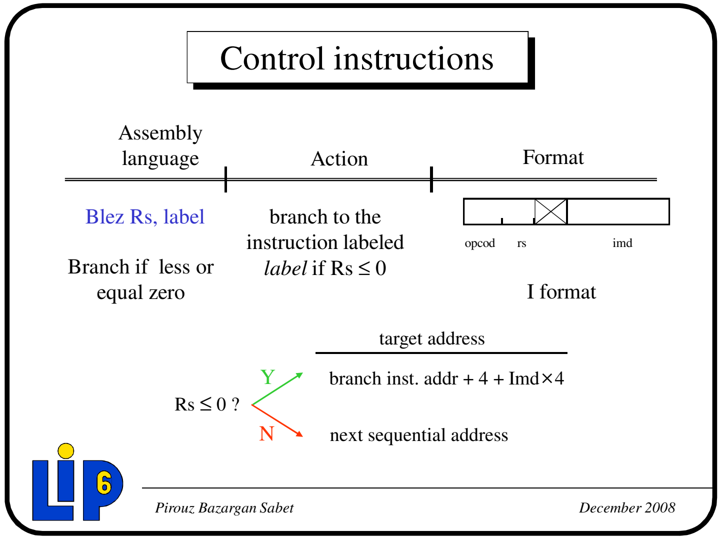
Assembly language	Action	Format
Bltz Rs, label Branch if less than zero	branch to the instruction labeled *label* if Rs < 0	opcod rs imd I format

target address

Rs < 0 ? Y → branch inst. addr + 4 + Imd×4

N → next sequential address

Pirouz Bazargan Sabet *December 2008*



Control instructions

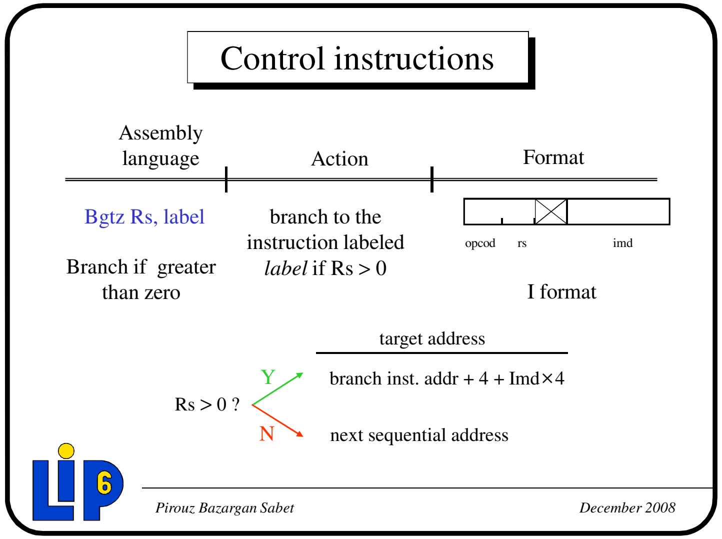
Assembly language	Action	Format
Blez Rs, label Branch if less or equal zero	branch to the instruction labeled *label* if Rs ≤ 0	opcod rs imd I format

target address

Rs ≤ 0 ? Y → branch inst. addr + 4 + Imd×4

N → next sequential address

Pirouz Bazargan Sabet *December 2008*



Control instructions

Assembly language	Action	Format
Bgtz Rs, label Branch if greater than zero	branch to the instruction labeled *label* if Rs > 0	opcod rs imd I format

target address

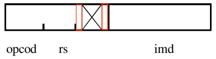
Rs > 0 ? Y → branch inst. addr + 4 + Imd×4

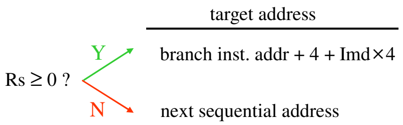
N → next sequential address

Pirouz Bazargan Sabet *December 2008*

Control instructions

Assembly language	Action	Format
Bgez Rs, label Branch if greater or equal zero	branch to the instruction labeled *label* if Rs ≥ 0	I format



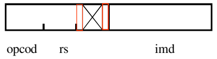


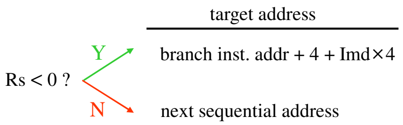
branch inst. addr + 4 + Imd×4

next sequential address

Control instructions

Assembly language	Action	Format
Bltzal Rs, label Branch if less than zero and link	branch to the instruction labeled *label* if Rs < 0 return address ➪ R31	I format



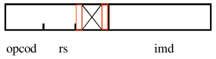


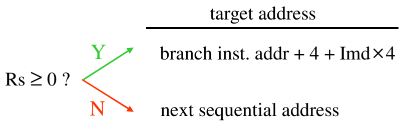
branch inst. addr + 4 + Imd×4

next sequential address

Control instructions

Assembly language	Action	Format
Bgezal Rs, label Branch if greater or equal zero and link	branch to the instruction labeled *label* if Rs ≥ 0 return address ➪ R31	I format





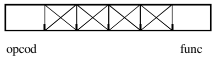
branch inst. addr + 4 + Imd×4

next sequential address

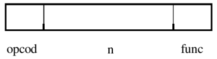
Instruction Set

- Computation instructions
- Memory access instructions
- Control instructions
- System instructions

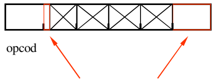
System instructions

Assembly language	Action	Format
Syscall System Call	Call a service of the operating system Stop executing the current program and jump to the Exception Handler	 R format

System instructions

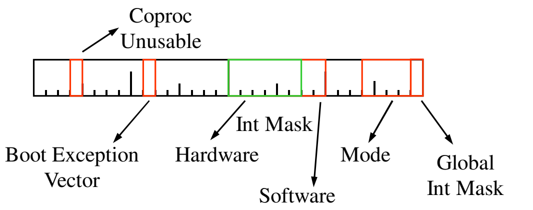
Assembly language	Action	Format
Break n Break	Call the debugger through the operating system Stop executing the current program and jump to the Exception Handler	 R format

System instructions

Assembly language	Action	Format
Eret Exception return	Return from exception Restore Status Register and jump to the interrupted program Exc if executed under User mode	 Bits # 25 and 0 to 5 used to extend the opcod I* format

System instructions

- Status Register



```
If    xx 1x => xx 0x
else        => xx 00
```

System instructions

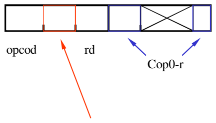
Assembly language | Action | Format

Mfc0 Rd, S

Move from coprocessor 0

Move a Coprocessor 0 register into an integer register

Exc if executed in User mode



Bits # 21 to 25 used to extend the opcod

I* format

System instructions

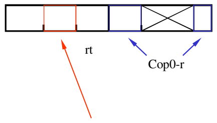
Assembly language | Action | Format

Mtc0 Rt, S

Move to coprocessor 0

Move an integer register into a Coprocessor 0 register

Exc if executed in User mode



Bits # 21 to 25 used to extend the opcod

I* format

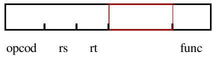
System instructions

Assembly language | Action | Format

Teq Rs, Rt

Trap if equal

Conditional call of a service of the operating system



R format

Stop executing the current program and jump to the Exception Handler

System instructions

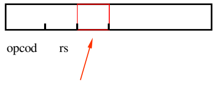
Assembly language | Action | Format

Teqi Rs, Imd

Trap if equal immediate

Conditional call of a service of the operating system

Imd is sign extended



Bits # 16 to 20 used to extend the opcod

I* format

Stop executing the current program and jump to the Exception Handler

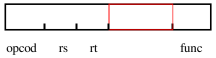
System instructions

Assembly language | Action | Format

Tne Rs, Rt

Trap if not equal

Conditional call of a service of the operating system



R format

Stop executing the current program and jump to the Exception Handler

System instructions

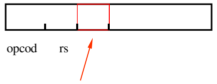
Assembly language | Action | Format

Tnei Rs, Imd

Trap if not equal immediate

Conditional call of a service of the operating system

Imd is sign extended



Bits # 16 to 20 used to extend the opcod

I* format

Stop executing the current program and jump to the Exception Handler

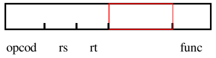
System instructions

Assembly language | Action | Format

Tlt Rs, Rt

Trap if less than

Conditional call of a service of the operating system



R format

Stop executing the current program and jump to the Exception Handler

System instructions

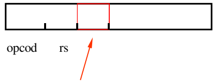
Assembly language | Action | Format

Tlti Rs, Imd

Trap if less than immediate

Conditional call of a service of the operating system

Imd is sign extended



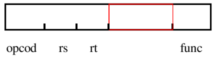
Bits # 16 to 20 used to extend the opcod

I* format

Stop executing the current program and jump to the Exception Handler

System instructions

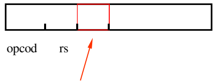
Assembly language	Action	Format
Tltu Rs, Rt Trap if less than unsigned	Conditional call of a service of the operating system	R format



Stop executing the current program and jump to the Exception Handler

System instructions

Assembly language	Action	Format
Tltiu Rs, Imd Trap if less than immediate unsigned	Conditional call of a service of the operating system Imd is sign extended	I* format

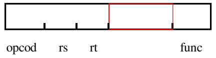


Bits # 16 to 20 used to extend the opcod

Stop executing the current program and jump to the Exception Handler

System instructions

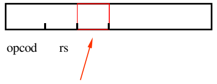
Assembly language	Action	Format
Tge Rs, Rt Trap if greater or equal	Conditional call of a service of the operating system	R format



Stop executing the current program and jump to the Exception Handler

System instructions

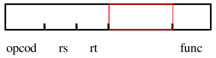
Assembly language	Action	Format
Tgei Rs, Imd Trap if greater or equal immediate	Conditional call of a service of the operating system Imd is sign extended	I* format



Bits # 16 to 20 used to extend the opcod

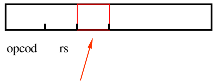
Stop executing the current program and jump to the Exception Handler

System instructions

Assembly language	Action	Format
Tgeu Rs, Rt Trap if greater or equal unsigned	Conditional call of a service of the operating system	 R format

Stop executing the current program and jump to the Exception Handler

System instructions

Assembly language	Action	Format
Tgeiu Rs, Imd Trap if greater or equal immediate unsigned	Conditional call of a service of the operating system Imd is sign extended	 Bits # 16 to 20 used to extend the opcod I* format

Stop executing the current program and jump to the Exception Handler