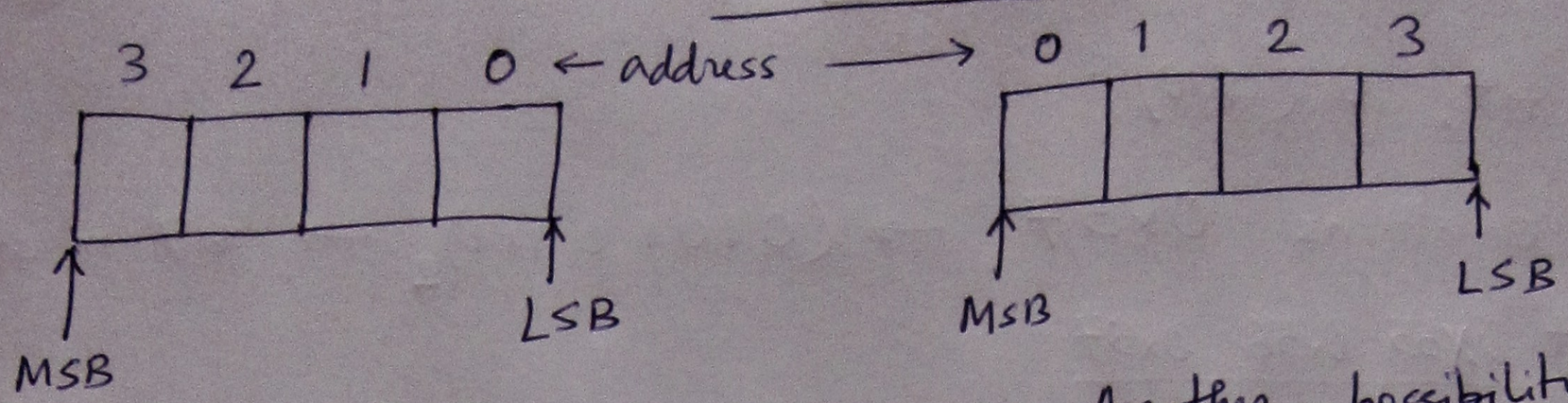
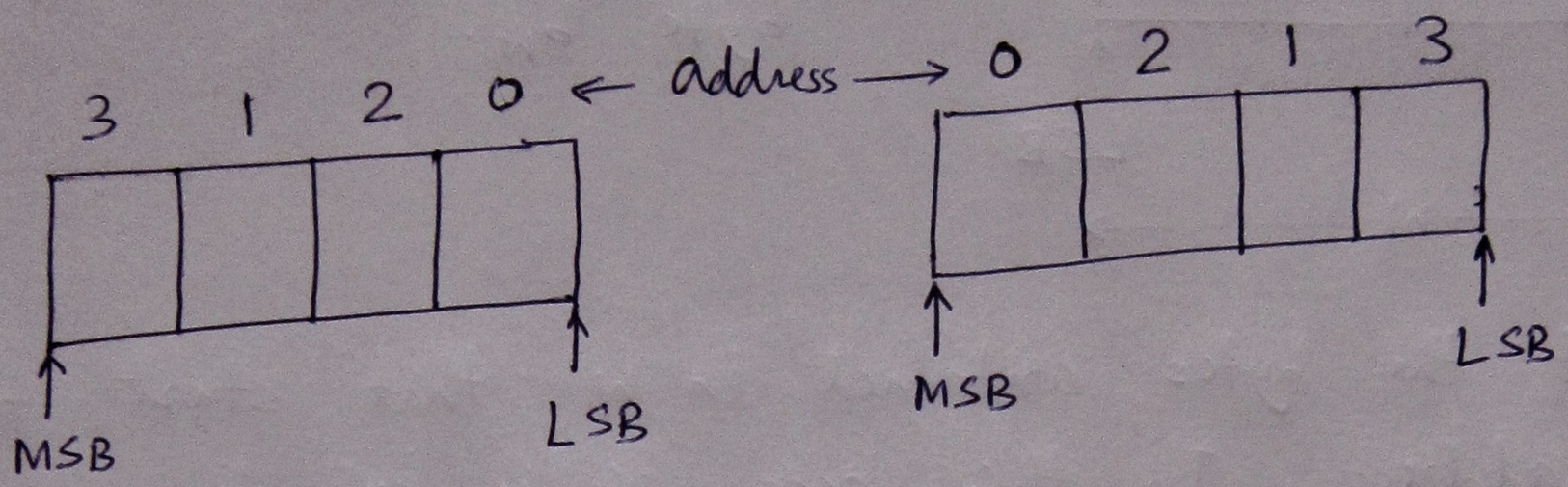
Slide-42



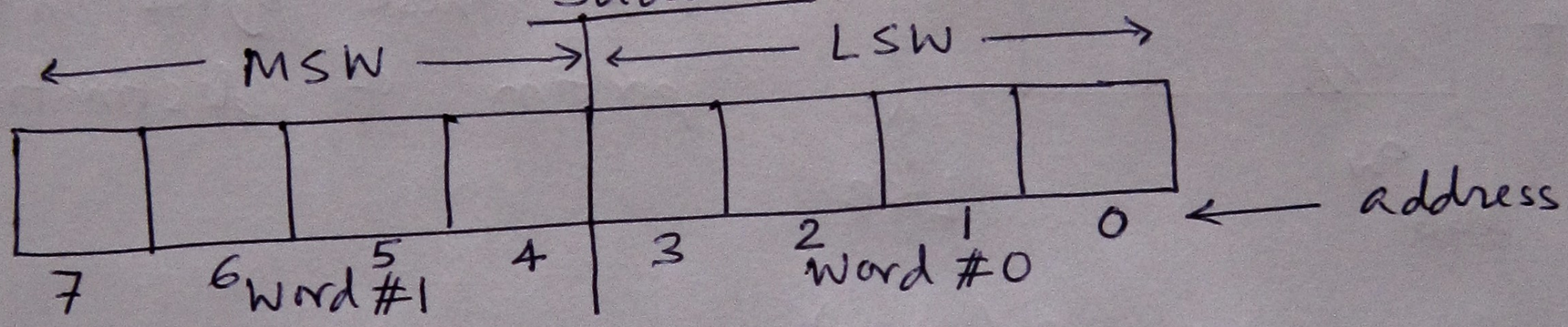
One possibility

Another possibility

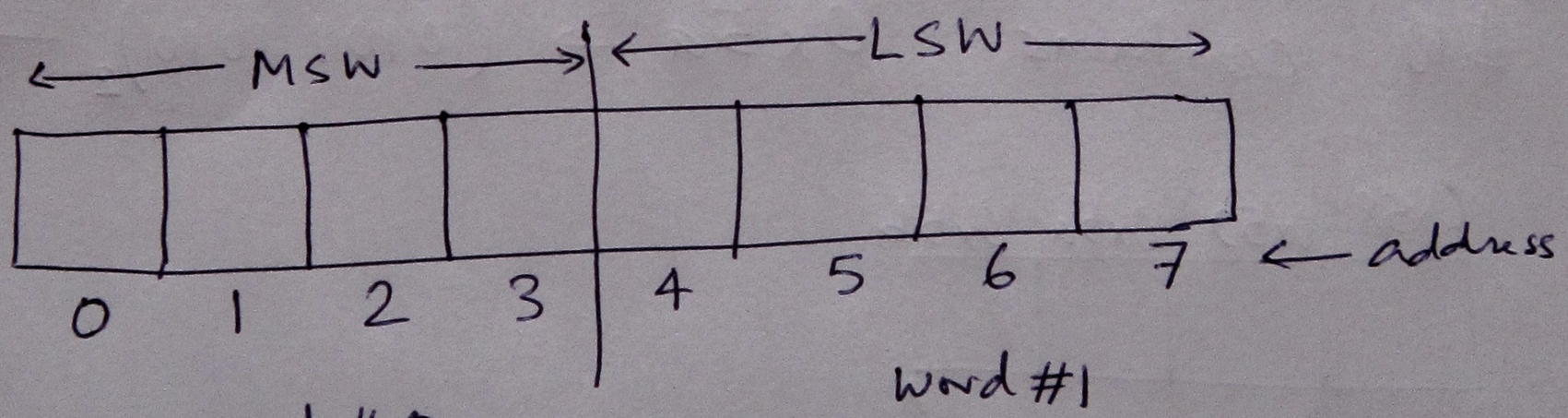


More possibilities

Slide-44



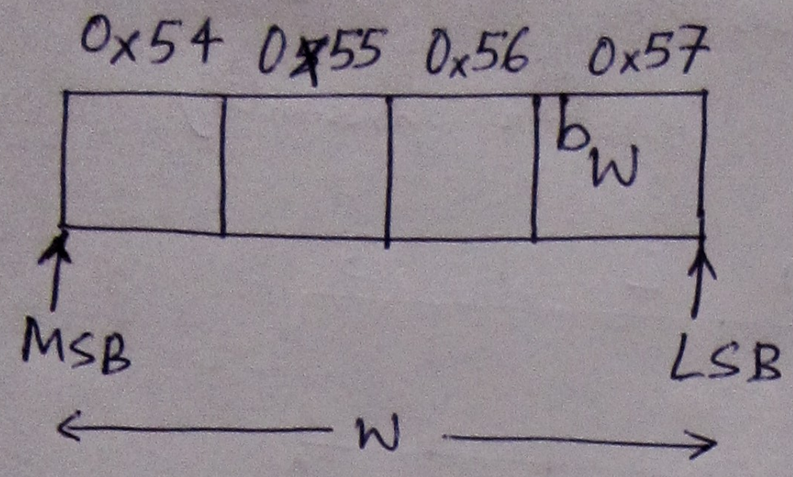
Little endian



Big endian

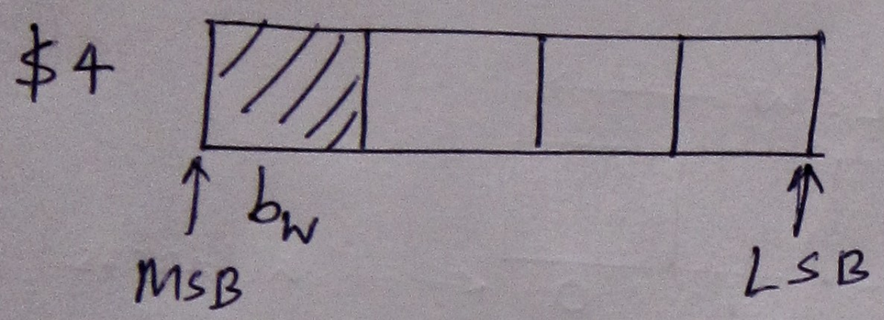
lwl $4, 0($10)

$10 cantains 0x57 → $(01010111)_2$ → 87 in decimal



the address point to the byte b_W.

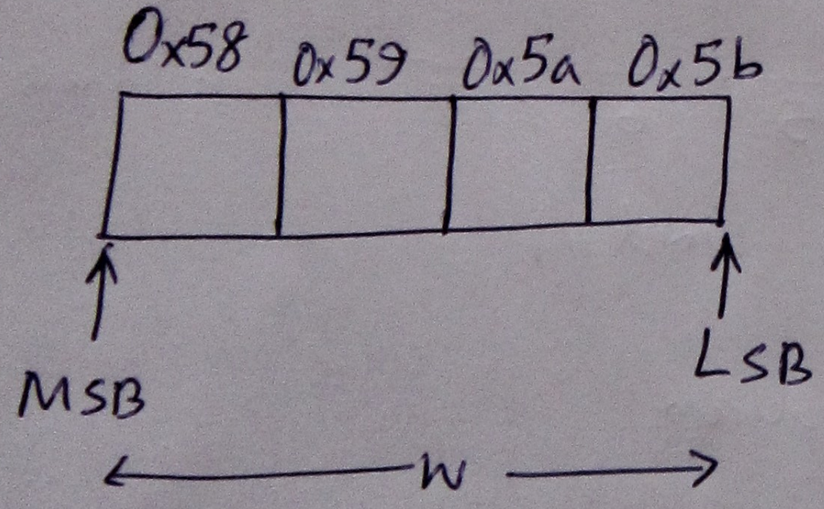
Extract the bytes contained in w that start from this address. In this case, it is just b_W.



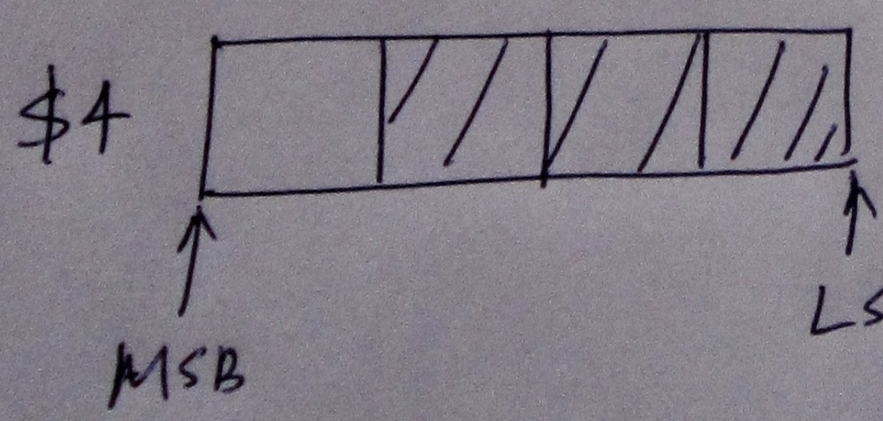
Put these bytes in the upper portion of destination register.

lwr $4, 0($10)

$10 cantains 0x5a → $(0101\ 1010)_2$ → 90 in decimal



Extract the bytes that end at this address starting from the beginning (i.e. left most) of w.



Put these bytes in the lower portion of destination register.

lwl and lwr together fetch unaligned words.

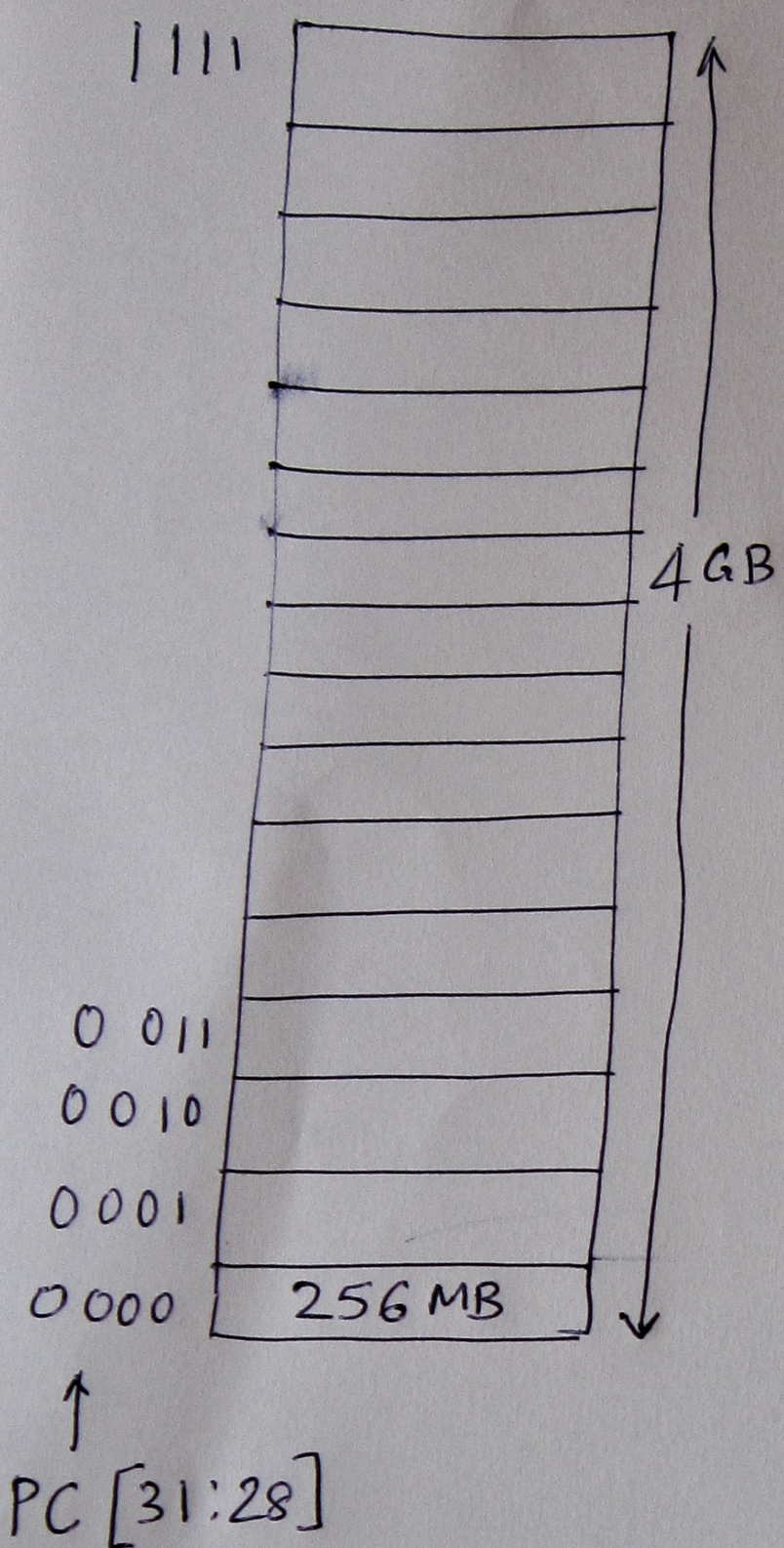
Slide-97

```
f(...) ← caller of g
{
   .
   .
   .
   g(...); ← caller of f
   .
   .
   .
}
```

```
g(....) ← caller of f
{
   .
   .
   .
}
```

Multiple functions can call g( ).

Slide - 144



Can jump anywhere within the current 256 MB region.

address of jump target

= {PC[31:28], target << 2}

address of conditional branch

target = PC + sign-ext (offset << 2)

Linking example

prog1.c :

```
int x; extern void B(int);
void A(int)
{
    uses x;
    B(x);
    :
}
```

prog2.c :

```
int Y; extern void A(int);
void B(int)
{
    uses Y;
    A(Y);
    :
}
```

gcc -c prog1.c -o prog1.o

gcc -c prog2.c -o prog2.o

We would like to link prog1.o and prog2.o.

Note : main function is not shown for brevity.

prog1.0

Header:

Name: procedure A

Text size: 0x100 (in bytes)

Data size: 0x20 (in bytes)

Text segment:

Address	Instruction
0	lw $a0, 0($gp)
4	jal 0
...	....

Data segment:

Address	Label
0	X
..	...

Relocation info:

Address	Instruction type	Dep.
0	lw	X
4	jal	B

Symbol table:

Label	Address
X	-
B	-

Need to find the addresses of X and B to fix the lw and jal instructions.

Prog 2.0 :

Header:

Name: Procedure B

Text size: 0x200

Data size: 0x30

Text segment:

Address	Instruction
0	sw $a1, 0($gp)
4	jal 0
...	....

Data segment:

0	Y
..	...

Relocation info:

Address	Instruction type	Dep.
0	sw	Y
4	jal	A

symbol table:

Label	Address
Y	-
A	-

Need to find the addresses of Y and A to fix the sw and jal instructions.

Text segment of an executable starts at address 0x400000. Please refer to the 32-bit MIPS memory map. Therefore, we can place procedure A starting at address 0x00400000. Since size of A is 0x100 bytes, we can start B at address 0x00400100. Accordingly, we can fix the jal instructions.

The global/static data segment starts at address 0x10000000. So, we can place the data of A starting at address 0x10000000. Since the data size of A is 0x20 bytes, we can place the data of B at 0x10000020. Recall that $gp is initialized to 0x10008000. So, we can fix the displacements of X and Y accordingly. The final executable header after linking is shown below.

Text size : 0x300 bytes

Data size : 0x50 bytes

Text segment:

Address	Instruction
0x00400000	lw $a0, 0x8000($gp)
0x00400004	jal 0x400100
...	...
0x00400100	sw $a1, 0x8020($gp)
0x00400104	jal 0x400000
...	...

Data segment:

Address	Data
0x10000000	X
...	...
0x10000020	Y
...	...