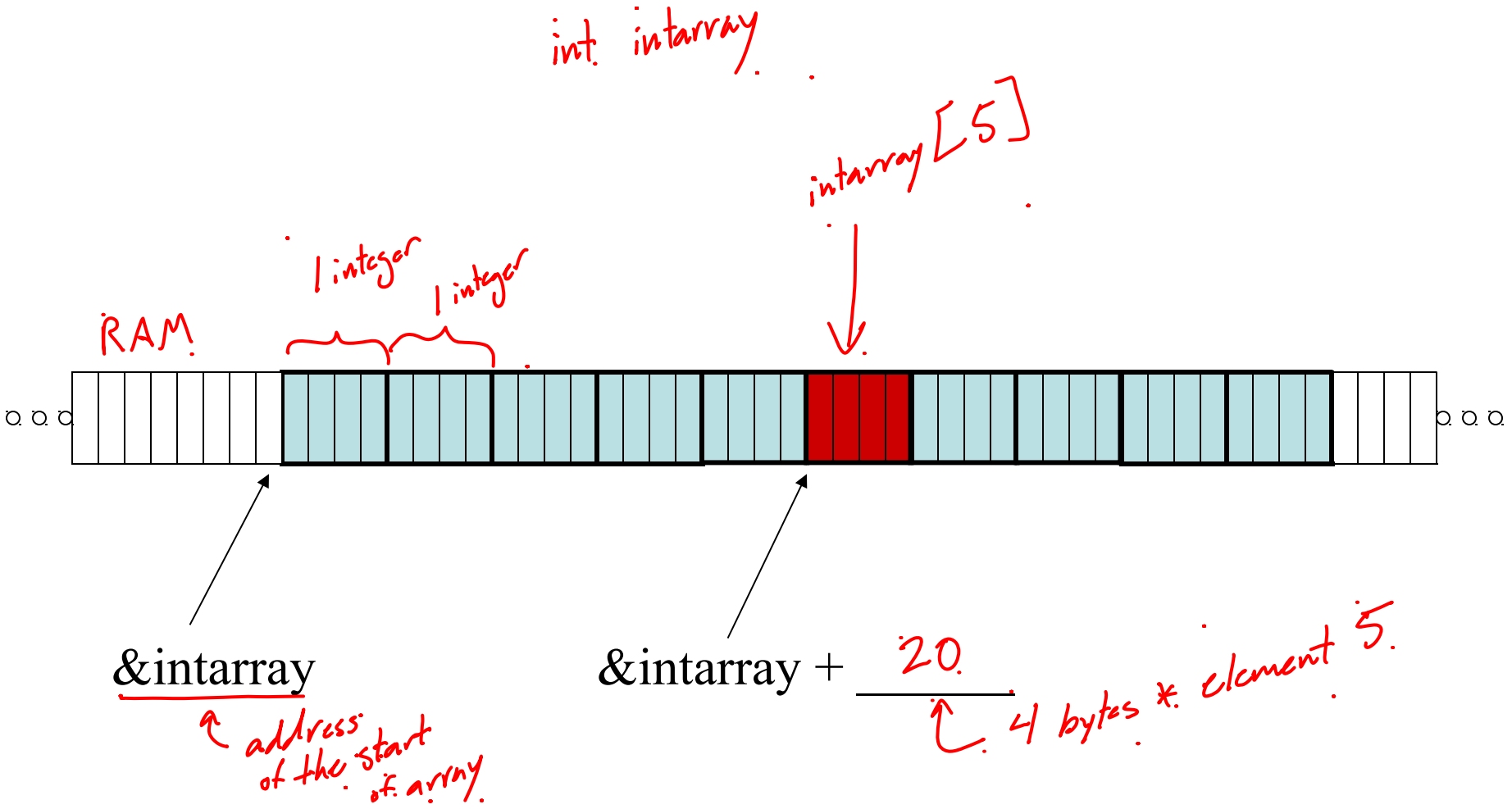
MIPS Registers – 32 registers

Name	Reg Number	Usage	Preserved across call?
$zero	0	The constant 0	Yes
$v0-$v1	2-3	Function results	No
$a0-$a3	4-7	Arguments	No
$t0-$t7	8-15	Temporaries	No
$s0-$s7	16-23	Saved	Yes
$t8-$t9	24-25	More temporaries	No
$gp	28	Global pointer	Yes
$sp	29	Stack pointer	Yes
$fp	30	Frame pointer	Yes
$ra	31	Return address	Yes

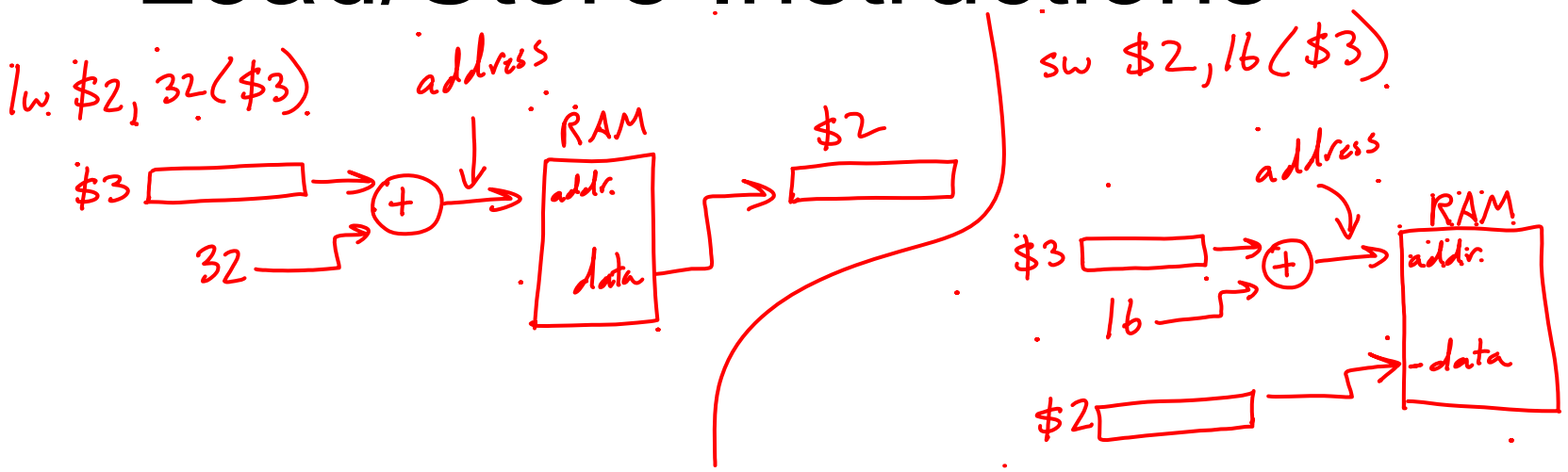
Memory Setup in C/Java



Words and bytes

- 4 bytes = 1 word = 32 bits
- Word addressable – each word can be accessed individually
- Byte addressable – each byte can be accessed individually

Load/Store Instructions



Operation	rs	rt	rd	shamt	funct	meaning
lw $2, 32($3) load word	3	2	← 32 →			$2 = M[32 +$3]
sw $2, 16($3) ↑ source	3	2	← 16 →			M[16 +$3] = $2
lb $2, 0($3) load byte	3					$2 = M[0 +$3]
sb $2, 3($3)	3					M[3 +$3] = $2

MIPS Example 2

Translate from C code
int A[100]; // ints are 4 bytes in Java/C
char B[100]; // chars are 1 byte in C

Assumptions:
&(A[0]) is in $s0,
&(B[0]) is in $s1
c is in $t1

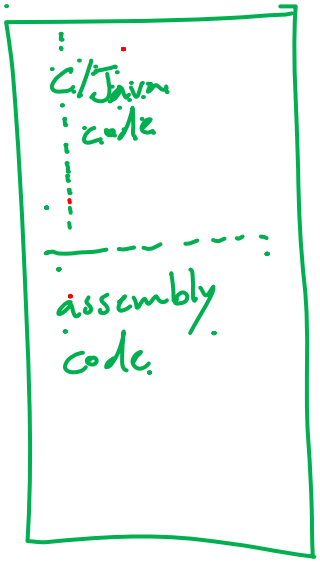
```
char c = B[50];
A[1] = A[5] + 7;
```

```
lb $t1, 50($s1)
lw $t0, 20($s0)
addi $t0, $t0, 7
sw $t0, 4($s0)
```

5×4

1×4

- may work with a partner
- spim = /home/jseng/bin/spim
- mipsdemo.asm
- due next Friday

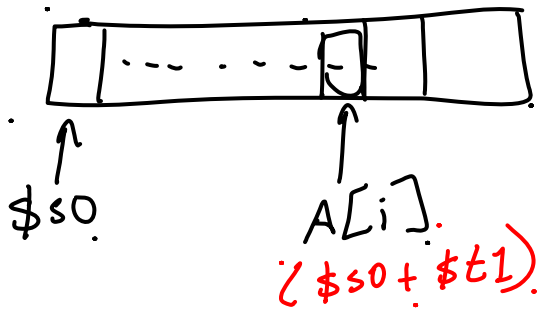


MIPS Example 3

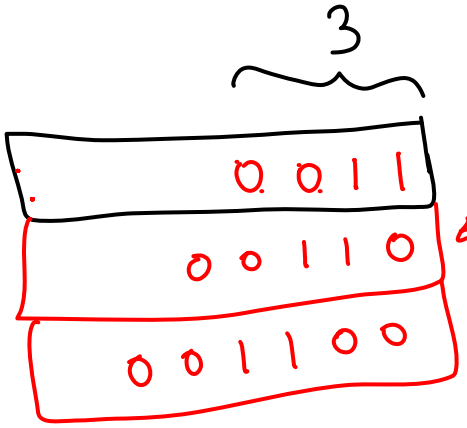
Read 2.1 - 2.8

Translate:

```
int A[100];
int i;
x = A[i];
```



Assumptions:
&(A[0]) is in $s0,
x is in $t0
i is in $s1




```
sll $t1, $s1, 2   <- shift left 2 places (*4)
add $t1, $t1, $s0
lw  $t0, 0($t1)
```

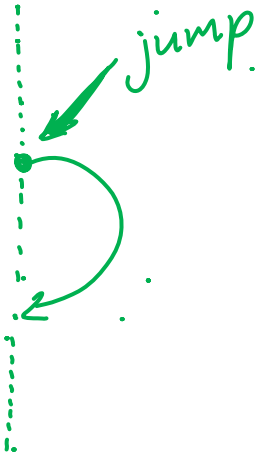
Loads/Stores

- lw $s0, 4($s1)

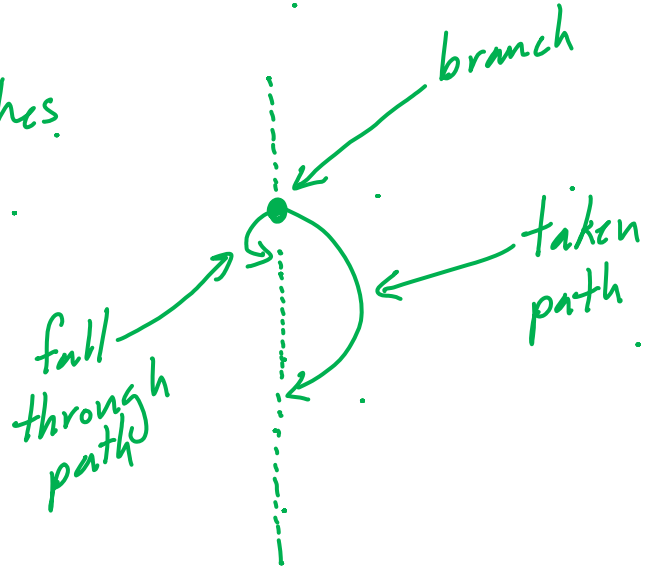
- sw $s0, 4($s1)

Control Operations

Unconditional Branches
- branch that is always taken



Conditional Branches
- branch based on a test



Control Operations

MIPS Example 4

```
sum=0;
for(i=0; i < 100; i++)
      sum += A[i];
```

•translated into more detail (with gotos)

```
            i = 0;
            sum = 0;
loop:       if ( i >= 100 )    goto end ;
            sum += A[i];
            i++                ;
            goto loop          ;
end:
```

Branches

j loop

- goto loop
- if (i >= 100) goto end