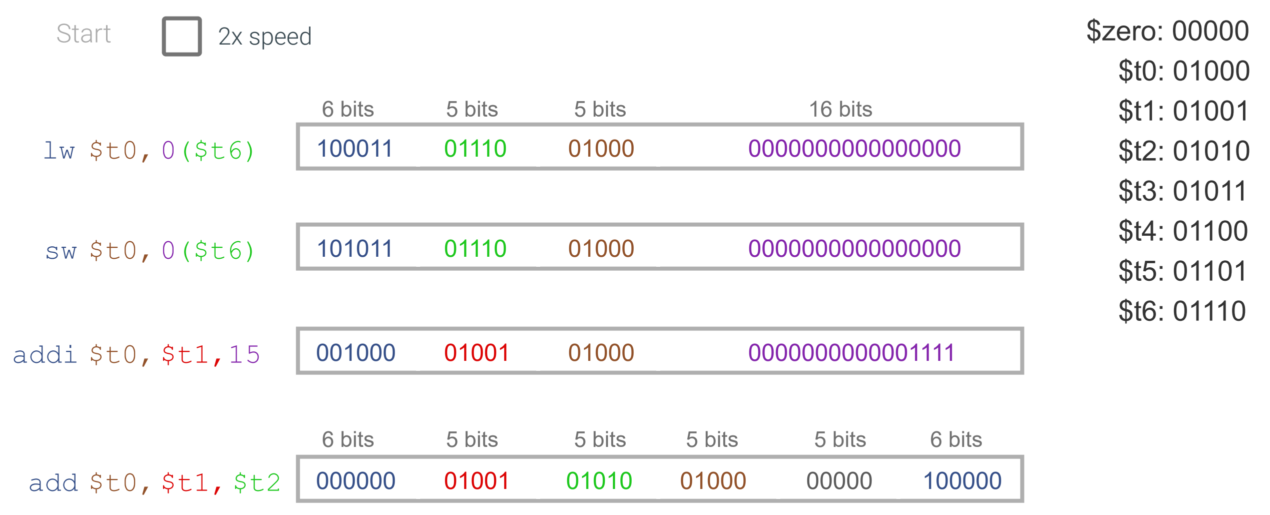
7.1 Machine instructions

Load, store, and add as 0s and 1s

A processor processes instructions in the form of 0's and 1's, each known as a ***machine instruction***. In MIPS, each machin
32 bits. Some bits, called the ***opcode*** ("operation code"), encode a machine instruction's operations like load word, store wo
machine instruction bits, each known as an ***operand***, indicate what register, address, or literal values are involved in an inst

PARTICIPATION ACTIVITY 7.1.1: Representing MIPS assembly instructions as machine instructions.

Start 2x speed



```
                     6 bits   5 bits   5 bits        16 bits
lw $t0, 0($t6)       100011   01110    01000    0000000000000000

sw $t0, 0($t6)       101011   01110    01000    0000000000000000

addi $t0, $t1, 15    001000   01001    01000    0000000000001111

                     6 bits   5 bits   5 bits   5 bits   5 bits   6 bits
add $t0, $t1, $t2    000000   01001    01010    01000    00000    100000
```

```
$zero: 00000
  $t0: 01000
  $t1: 01001
  $t2: 01010
  $t3: 01011
  $t4: 01100
  $t5: 01101
  $t6: 01110
```

PARTICIPATION ACTIVITY 7.1.2: Machine instructions.

Refer to the above assembly and machine instructions.

1) lw's opcode is _____ .

- ○ 000000
- ○ 100011
- ○ 101001

2) Registers like $t0 or $t6 are encoded using how many bits?

- ○ 5
- ○ 6

3) Register $t0 is encoded as _____ .

- ○ 00000
- ○ 01000
- ○ 01110

4) In the lw machine instruction, the second field represents the _____ .

- ○ destination register
- ○ immediate
- ○ base address

5) In the addi machine instruction, the immediate is represented by how many bits?

- ○ 5
- ○ 6
- ○ 16

6) How many bits are used to denote that an instruction should perform an add?

- ○ 6
- ○ 12

7) In the add machine instruction, the second field is the destination register.

- ○ True
- ○ False

PARTICIPATION ACTIVITY 7.1.3: Translating an lw assembly instruction to a MIPS machine instruction.

Finish translating this assembly instruction to a machine instruction:

```
lw $t6, 0($t2)
__a___  __b__  __c__  0000000000000000
```

1) a

Check Show answer

2) b

Check Show answer

3) c

Check Show answer

PARTICIPATION ACTIVITY 7.1.4: Translating an add assembly instruction to a MIPS machine instruction.

Finish translating this assembly instruction to a machine instruction:

```
add $t6, $t5, $t4
__a___  __b__  __c__  __d__ 00000 __e___
```

1) a

Check Show answer

2) b

Check Show answer

3) c

Check Show answer

4) d

Check Show answer

5) e

Check Show answer

MIPSzy machine instructions

The table below shows MIPSzy's register encodings. The subsequent table shows all of MIPSzy's machine instructions, wit blue, registers in orange, green, and red, immediate values in purple, and unused bits in grey.

Table 7.1.1: MIPSzy register encodings.

Name	$zero	$t0	$t1	$t2	$t3	$t4	$t5	$t6
Encoding	00000	01000	01001	01010	01011	01100	01101	01110

Table 7.1.2: MIPSzy machine instructions.

Assembly	Machine
`lw $t0, 0($t1)`	100011 01001 01000 0000000000000000
`sw $t0, 0($t1)`	101011 01001 01000 0000000000000000
`addi $t0, $t1, 15`	001000 01001 01000 0000000000001111
`add $t0, $t1, $t2`	000000 01001 01010 01000 00000 100000
`sub $t0, $t1, $t2`	000000 01001 01010 01000 00000 100010
`mult $t1, $t2`	000100 01001 01010 00000 00000 011000

mflo $t0	000000 00000 00000 01000 00000 010010
beq $t1, $t2, BLabel	000100 01001 01010 0000000000000010
bne $t1, $t2, BLabel	000101 01001 01010 0000000000000010
slt $t0, $t1, $t2	000000 01001 01010 01000 00000 101010
j JLabel	000010 00000000000000000000000101
jal JLabel	000011 00000000000000100000000101
jr $t1	000000 01001 00000 00000 00000 001000

Assume BLabel becomes an immediate of 2, and JLabel 5. Creating immediates for branches/jumps is in another section.

$t0, $t1, and $t2 are used for registers. Other registers could be used.

addi's immediate value is shown as 15. That value is arbitrary.

PARTICIPATION ACTIVITY 7.1.5: MIPSzy machine instructions.

1) Different MIPSzy instructions have different numbers of bits.

- True
- False

2) addi uses _____ bits for the immediate value.

- 5
- 16

3) For a sub instruction, the first 6 bits are 000000, and the last 6 bits are _____ .

- ○ 100000
- ○ 100010

4) add and sub make use of all 32 bits.

- ○ True
- ○ False

5) For an slt instruction, the first 6 bits are 000000, and the last 6 bits are _____ .

- ○ 101010
- ○ 000000

6) For a j instruction, the immediate is _____ bits.

- ○ 26
- ○ 32

PARTICIPATION ACTIVITY 7.1.6: Translating an addi instruction to a MIPS machine instruction.

Finish translating this assembly instruction to a machine instruction:

```
addi $t3, $t4, 7
__a___  __b__  __c__  0000000000___d___
```

1) a

Check Show answer

2) b

Check Show answer

3) c

Check Show answer

4) d (just the last 6 bits of the immediate field).

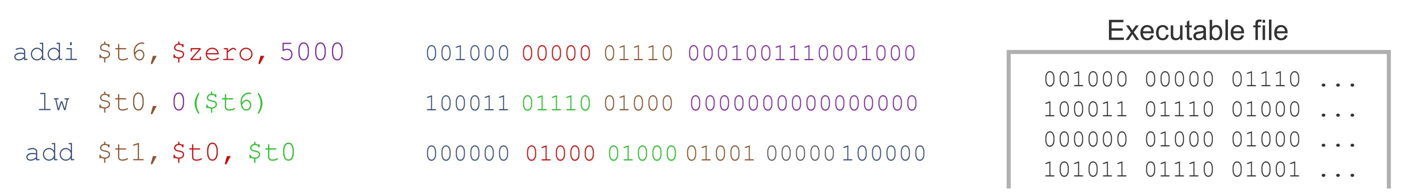
Check Show answer

Executable files

Assembly instructions can be translated to machine instructions. An ***executable file*** contains the 0's and 1's of a program's instructions and can be loaded into an instruction memory and then executed (run). The 0's and 1's are placed into a proce memory, and the processor then executes each machine instruction.

PARTICIPATION ACTIVITY 7.1.7: Assembly instructions can be translated to machine instructions (0's and 1's) representing an executable file.

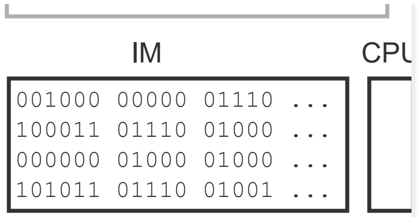
Start 2x speed



```
addi $t6, $zero, 5000     001000 00000 01110 0001001110001000
 lw  $t0, 0($t6)          100011 01110 01000 0000000000000000
 add $t1, $t0, $t0        000000 01000 01000 01001 00000 100000
```

```
001000 00000 01110 ...
100011 01110 01000 ...
000000 01000 01000 ...
101011 01110 01001 ...
```

```
sw  $t1, 0($t6)          101011 01110 01001 0000000000000000
```



```
001000 00000 01110 ...
100011 01110 01000 ...
000000 01000 01000 ...
101011 01110 01001 ...
```

PARTICIPATION ACTIVITY 7.1.8: Executable files.

1) An assembly program is placed into a processor's instruction memory.

○ True

○ False

2) An executable file is convenient for humans to read.

○ True

○ False

3) A processor executes each machine instruction one at a time (conceptually).

○ True

○ False

Provide feedback on this section