MIPS Instructions, Memory Storage

General Purpose Registers

REGISTER NAME	REGISTER #	DESCRIPTION
$zero	$0	a special purpose register which always contains a constant value of 0. It can be read, but cannot be written.
$at	$1	a register reserved for the assembler. If the assembler needs to use a temporary register (e.g. for pseudo instructions), it will use $at, so this register is not available for use programmer use.
$v0-$v1	$2-$3	registers are normally used for return values for subprograms. $v0 is also used to input the requested service to syscall.
$a0-$a3	$4-$7	registers are used to pass arguments (or parameters) into subprograms.
$t0-$t9	$8-$15, $24-$25	registers are used to store temporary variables. The values of temporary variables can change when a subprogram is called.
$s0-$s7	$16-$23	registers are used to store saved values. The values of these registers are maintained across subprogram calls.
$k0-$k1	$26-$27	registers are used by the operating system, and are not available for use programmer use.
$gp	$28	pointer to global memory. Used with heap allocations.
$sp	$29	stack pointer, used to keep track of the beginning of the data for this method in the stack.
$fp	$30	frame pointer, used with the $sp for maintaining information about the stack. This text will not use the $fp for method calls.
$ra	$31	return address, a pointer to the address of the instruction to execute when returning from a subprogram.

Source: Introduction To MIPS Assembly Language Programming by Charles W. Kann, 2015

Special Registers

pc	
hi	After a mult instruction - After a div instruction
lo	After a mult instruction - After a div instruction

Note

You may use either the name or register number when writing a program.

Example

Assembler Directives

Indicated by a _____

_____

._____ - _____

._____ - _____

Data Directives

.space

Description

General Format

Stored As

ADDRESS	CONTENTS	
label + 4		
label + 3		
label + 2		
label + 1		
label + 0		

Example Usage

.ascii

Description

General Format

Stored As

ADDRESS	CONTENTS	
label + 2		
label + 1		
label + 0		

Example Usage

.asciiz

Description

General Format

Stored As

ADDRESS	CONTENTS	
label + 4		
label + 3		
label + 2		
label + 1		
label + 0		

Example Usage

.byte

Description

General Format

Stored As

ADDRESS	CONTENTS	
label + 5		
label + 4		
label + 3		
label + 2		
label + 1		
label + 0		

Example Usage

.half

Description

General Format

Stored As

ADDRESS	CONTENTS	
label + 5		
label + 4		
label + 3		
label + 2		
label + 1		
label + 0		

Example Usage

.word

Description

General Format

Stored As

ADDRESS	CONTENTS	
label + 7		
label + 6		
label + 5		
label + 4		
label + 3		
label + 2		
label + 1		
label + 0		

Example Usage

.float

Description	Stored As
General Format	

ADDRESS	CONTENTS	
label + 7		
label + 6		
label + 5		
label + 4		
label + 3		
label + 2		
label + 1		
label + 0		

Example Usage

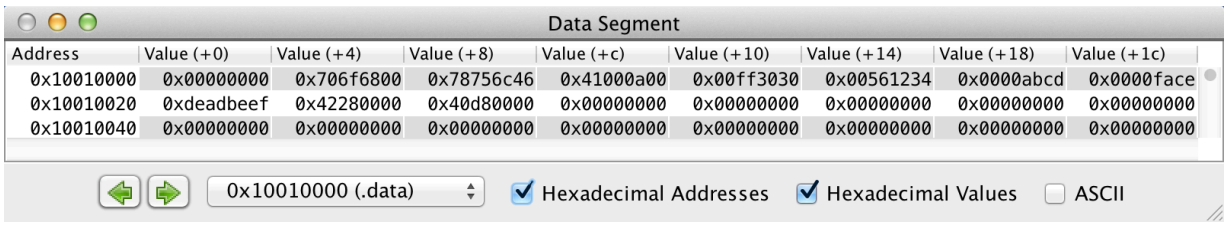
42: 0x42280000 6.75: 0x40D80000

Example

(an expanded version of this example can be found here)

```
.data

.space   5
.ascii   "hop"
.asciiz  "Flux"
.byte    10 0x00 0x41 48 0x30 0xFF
.half    0x1234 0x56 0xABCD
.word    0xFACE 0xDEADBEEF
.float   42 6.75
```



Labels

Converted by the assembler to an _____

Indicated by a _____

Used to label _____ and _____

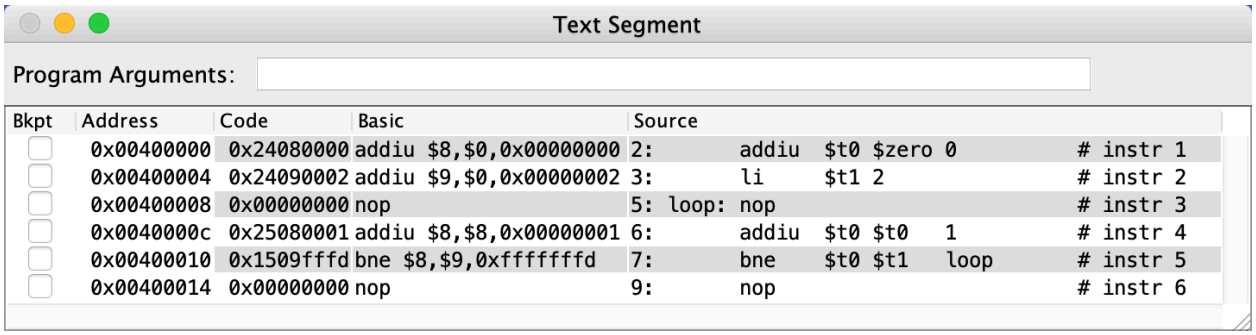
Can be used in _____ and _____segments of code.

Sample Program

```
.text
      addiu  $t0 $zero 0          # instr 1
      li     $t1 2                # instr 2

loop: nop                         # instr 3
      addiu  $t0 $t0   1          # instr 4
      bne    $t0 $t1   loop       # instr 5

      nop                         # instr 6
```



instruction address - machine code - native commands - written by programmer

MIPS Address Space (Not to Scale)

0x9000 0000	MMIO
	Kernel Data
0x7fff fe00	Stack
	Heap
0x1004 0000	
0x1001 0000	Static Data
	Program Text
0x0040 0000	
0x0000 0000	Reserved

The label loop represents which address?

Instruction Memory

MIPS ISA is _____

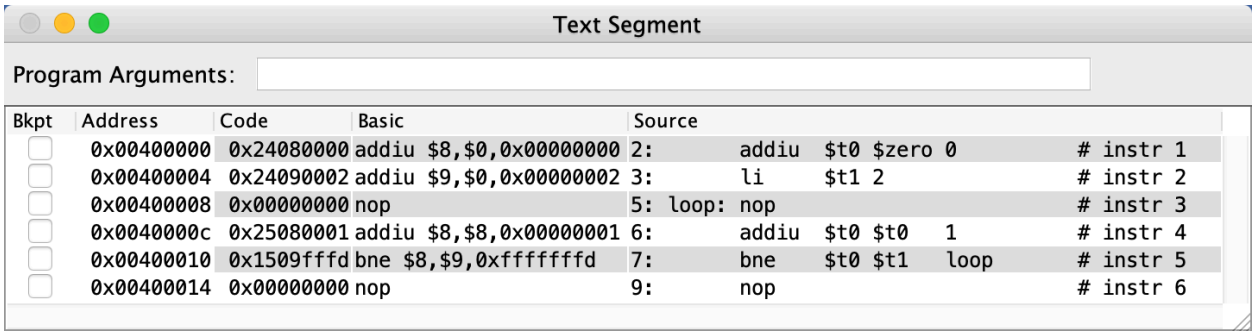
_____

A program is comprised of _____

which are encoded as _____(_____)

Therefore, each instruction takes up _____ memory locations.

Example



What is stored at each of these addresses?

ADDRESS	DATA (ENCODED INSTR)
0x0040 0014	
0x0040 0010	
0x0040 000C	
0x0040 0008	
0x0040 0004	
0x0040 0000	

A closer look just at addresses 0x00400000 through 0x00400007

ADDRESS	DATA (ENCODED INSTR)
0x0040 0007	
0x0040 0006	
0x0040 0005	
0x0040 0004	
0x0040 0003	
0x0040 0002	
0x0040 0001	
0x0040 0000	

Instruction Format

Many instructions use ___ registers: 1 _____ & 2 _____

Write the general form of an instruction: Write an example of an instruction:

Instruction Types

1 - _____

_____

2 - _____

_____

3 - _____

_____

Operate Instructions

Uses _____ unless unsigned is indicated.

e.g. _____

Example: Assume $t0 = 0xFFFFFFFF, $t1 = 0x00000001

`add $t2 $t0 $t1` Is there overflow?	`addu $t2 $t0 $t1` Is there overflow?

Register vs. Immediate Instructions

e.g. _____

`add $t2 $t1 $t0`	`addi $t2 $t1 0x4`

In MIPS, the immediate value can be _____ bits max.

Other Operate Instructions

e.g. _____

Loads and Stores

Load Word

Loads _____ of data

from memory into a _____

What is the value of $t0?

```
LI   $t1 0x1010
LW   $t0 ($t1)
```

$t0 = 0x _____ _____ _____ _____

ADDRESS	CONTENTS
0x1013	0xC0
0x1012	0xFF
0x1011	0xEE
0x1010	0xEE

Store Word

Takes a _____

and stores the value in _____

Example

What does memory look like after these instructions?

```
LI   $t1 0x1010
LI   $t0 0xABCDEF00
SW   $t0 ($t1)
```

ADDRESS	CONTENTS	
	BEFORE	AFTER
0x1013	0xC0	
0x1012	0xFF	
0x1011	0xEE	
0x1010	0xEE	

Load Half Word

Loads a _____(_____)

from memory into a _____

Uses _____

Example

What is the value of $t0?

```
LI   $t1 0x1010
LH   $t0 ($t1)
```

$t0 =

_____ _____ _____ _____

$t0 = 0x _____ _____ _____ _____

ADDRESS	CONTENTS
0x1013	0xFE
0x1012	0xED
0x1011	0xBA
0x1010	0xBE

Store Half Word

Stores the _____

_____(_____)

from a register into _____

Example

What does memory look like after these instructions?

```
LI   $t1 0x1010
ADDI $t0 $zero 0xEF00
SH   $t0 ($t1)
```

ADDRESS	CONTENTS	
	BEFORE	AFTER
0x1013	0xFE	
0x1012	0xED	
0x1011	0xBA	
0x1010	0xBE	

Load Byte

Loads a _____

from memory into a _____

Uses _____

Example

What is the value of $t0?

```
LI   $t1 0x1010
LB   $t0 ($t1)
```

$t0 = _____ _____ _____ _____

$t0 = 0x _____ _____ _____ _____

ADDRESS	CONTENTS
0x1013	0xFE
0x1012	0xED
0x1011	0xBA
0x1010	0xBE

Store Byte

Stores the _____

from a register into _____

Example

What does memory look like after these instructions?

```
LI   $t1 0x1010
ADDI $t0 $zero 0xABCD
SB   $t0 ($t1)
```

ADDRESS	CONTENTS	
	BEFORE	AFTER
0x1013	0xFE	
0x1012	0xED	
0x1011	0xBA	
0x1010	0xBE	

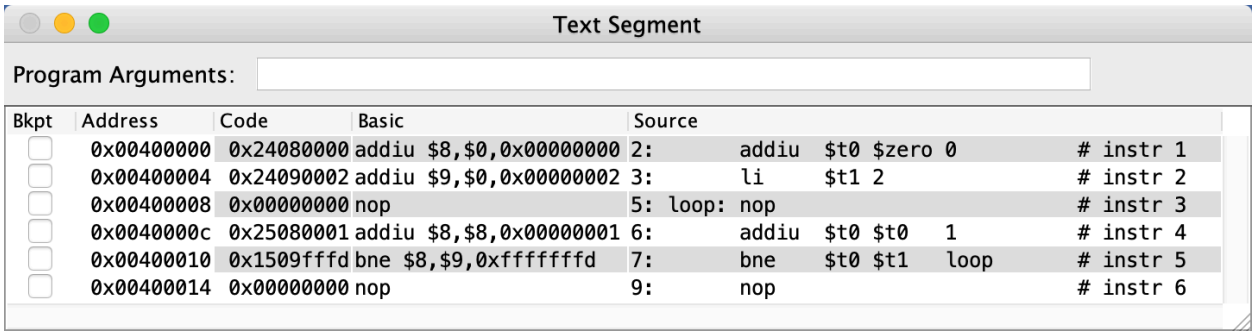
Unsigned Loads

e.g. _____

Control Flow Instructions

e.g. _____

Example



Instruction Execution Order

Assume each instruction takes 1 clock cycle to execute.

CLOCK CYCLE	PC	INSTRUCTION
1		
2		
3		
4		
5		
6		
7		
8		
9		

Example

Pseudocode	Assembly code

Endian-ness

The storage order of bytes in memory is determined by the _____

Example

```
LI   $t1  0x1014
LI   $t0  0x12345678
SW   $t0 ($t1)
```

How should we store the bytes in memory?

ADDRESS	CONTENTS
0x1017	
0x1016	
0x1015	
0x1014	

Little Endian

_____ stored at the

_____ memory address

Note

The order of

_____ within each

_____ stays the same.

Example

```
LI   $t1  0x1014
LI   $t0  0xABCDEF00
SW   $t0 ($t1)
```

$t0 = 0x _____ _____ _____ _____

ADDRESS	CONTENTS
0x1017	
0x1016	
0x1015	
0x1014	

Big Endian

_____ stored at the

_____ memory address

Note
The order of

_____ within each

_____ stays the same.

Example

```
LI   $t1  0x1014
LI   $t0  0xABCDEF00
SW   $t0 ($t1)
```

ADDRESS	CONTENTS
0x1017	
0x1016	
0x1015	
0x1014	

Examples

```
Assuming little endian storage format:
LI   $t1 0x1010
LW   $t0 ($t1)

$t0 = 0x ______  ______  ______  ______

Assuming big endian storage format:
LI   $t1 0x1010
LW   $t0 ($t1)

$t0 = 0x ______  ______  ______  ______
```

ADDRESS	CONTENTS
0x1015	0x34
0x1014	0x12
0x1013	0xFE
0x1012	0xED
0x1011	0xBA
0x1010	0xBE

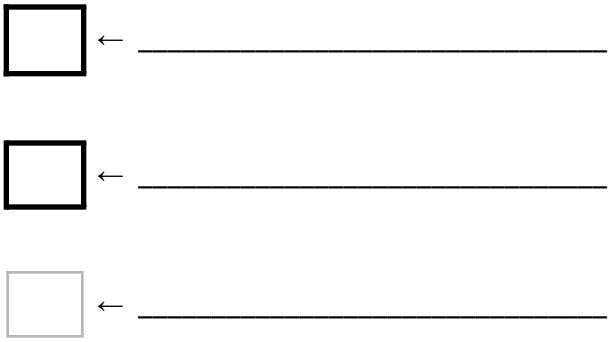
Note

MIPS uses _____ memory storage format.

```
What happens if we try to execute:
LI   $t1 0x1011
LW   $t0 ($t1)
```

Alignment

ADDRESS	CONTENTS
0x101C	0x11
0x101B	0xEF
0x101A	0xCD
0x1019	0xAB
0x1018	0x90
0x1017	0x78
0x1016	0x56
0x1015	0x34
0x1014	0x12



MIPS permits only

_____

memory accesses for

_____

How do we determine if an access is memory aligned?

For word alignment, look at _____

For half word alignment, look at _____

Which of the following memory accesses are aligned?

```
LW  $t0 (0x1011)                SH  $t0 (0x1011)
LW  $t0 (0x1018)                SH  $t0 (0x1019)
SW  $t0 (0x1014)                LB  $t0 (0x1011)
SW  $t0 (0x1022)                LBU $t0 (0x1014)
LH  $t0 (0x1011)                SB  $t0 (0x1015)
LHU $t0 (0x1013)                SB  $t0 (0x1017)
```

What happens if we try to make an unaligned memory access?

Pseudo Instructions (aka _____)

e.g. _____

Some instructions have several formats.

e.g. _____

Instruction Encoding

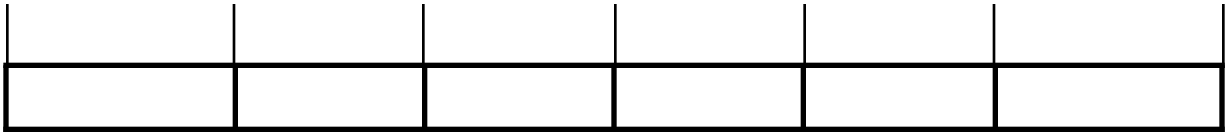
MIPS instructions are encoded in _____ bits.

There are _____ different instruction formats in MIPS.

How many general purpose registers does MIPS have? _____

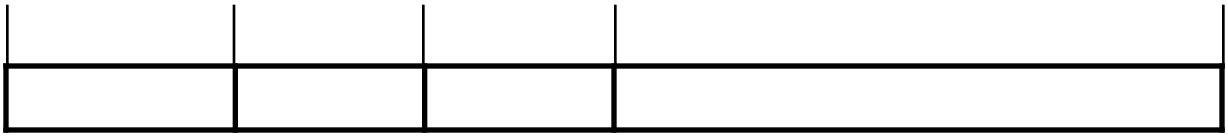
So, how many bits do we need for the registers? _____

_____ *type (_____ type)*



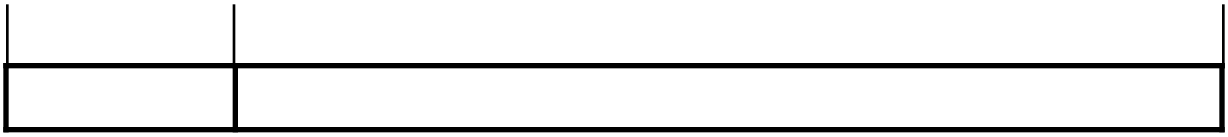
e.g. _____

_____ *type (_____ type)*



e.g. _____

_____ *type (_____ type)*



e.g. _____

Example: What types of instructions are the following instructions?

_____ ADD	_____ LI	_____ BLEZ
_____ BEQ	_____ SH	_____ J
_____ ADDI	_____ LB	_____ JR

How do pseudo ops get encoded as machine code?

Examples

Decode this instruction: 0x00AF8020

Decode this instruction: 0x2402000A

Decode this instruction: 0x00000000