MIPS Addressing Modes

Data Movement Instructions

In these instructions, assume rs is a register that contains an address.

INSTRUCTION	SYNTAX	DESCRIPTION
Load Word		➔ Go to the memory address stored in rs. ➔ Get 4 bytes of data, starting from the memory address in rs ➔ Load data into register rt ➔ Register rs must be word aligned
Load Half		➔ Go to the memory address stored in rs. ➔ Get 2 bytes of data, starting from the memory address in rs ➔ Load data into least significant 2 bytes of register rt ➔ Sign extend to fill the rest of the bits in register rt ➔ Register rs must be half word aligned
Load Half Unsigned		➔ Go to the memory address stored in rs. ➔ Get 2 bytes of data, starting from the memory address in rs ➔ Load data into least significant 2 bytes of register rt ➔ Zero extend to fill the rest of the bits in register rt ➔ Register rs must be half word aligned
Load Byte		➔ Go to the memory address stored in rs. ➔ Get 1 byte of data ➔ Load data into least significant byte of register rt ➔ Sign extend to fill the rest of the bits in register rt
Load Byte Unsigned		➔ Go to the memory address stored in rs. ➔ Get 1 byte of data ➔ Load data into least significant byte of register rt ➔ Zero extend to fill the rest of the bits in register rt

INSTRUCTION	SYNTAX	DESCRIPTION
Store Word		➔ Store data in rt starting at memory address rs ➔ Register rs must be word aligned
Store Half		➔ Store least significant 2 bytes of data in rt starting at memory address rs ➔ Register rs must be half word aligned
Store Byte		➔ Store least significant byte of data in rt starting at memory address rs

Addressing Modes

Ways of accessing operands

1 - _____

2 - _____

3 - _____

4 - _____

5 - _____

Register Direct

Operands are in registers.

Syntax: _____

e.g. _____

Immediate

Operand is a constant in the instruction.

Syntax: _____

e.g. _____

_____

Register Indirect / Base + Offset / Displacement

Register Indirect

Memory address is contained in rs.

Syntax: _____

What is the value of $t0?

```
ADDIU $t1  $zero 0x1010
LW    $t0 ($t1)
```

$t0 = 0x _____ _____ _____ _____

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

What is the value of $t0?

```
ADDIU $t6   $zero 0x1010
LH    $t0  ($t6)
```

$t0 =

_____ _____ _____ _____

$t0 = 0x _____ _____ _____ _____

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

Base + Offset / Displacement

Syntax: _____

Compute _____ by summing

_____

Effective address - _____

_____

e.g. _____

Example, Base + Offset/Displacement

What is the value of $t0?

```
ADDIU $t3    $zero 0x1010
LH    $t0  2($t3)

effective address: _____________________

_____________________________________

$t0 =

________  ________  ________  ________


$t0 = 0x _______  _______  _______  _______
```

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

What is the value of $t0?

```
ADDIU $t7    $zero 0x1014
LHU   $t0 -4($t7)

effective address: _____________________

_____________________________________


$t0 =

________  ________  ________  ________


$t0 = 0x _______  _______  _______  _______
```

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

What does memory look like after these instructions?

```
ADDIU $t5    $zero 0x1015
ADDIU $t0    $zero 0x1234
SH    $t0 -3($t5)

effective address: _____________________

_____________________________________

Is there a memory alignment error?
```

ADDRESS	CONTENTS	
	BEFORE	AFTER
0x1015	0x87	
0x1014	0x65	
0x1013	0xFE	
0x1012	0xED	
0x1011	0xBA	
0x1010	0xBE	

What does memory look like after these instructions?

```
ADDIU $t2    $zero 0x1012
ADDIU $t0    $zero 0xEFAA
SB    $t0  1($t2)

effective address: _____________________

_____________________________________

Is there a memory alignment error?
```

ADDRESS	CONTENTS	
	BEFORE	AFTER
0x1015	0x87	
0x1014	0x65	
0x1013	0xFE	
0x1012	0xED	
0x1011	0xBA	
0x1010	0xBE	

What does memory look like after these instructions?

```
ADDIU $t4    $zero 0x1015
ADDIU $t0    $zero 0x1234
SW    $t0 -3($t4)

effective address: _____________________

_____________________________________

Is there a memory alignment error?
```

ADDRESS	CONTENTS	
	BEFORE	AFTER
0x1015	0x87	
0x1014	0x65	
0x1013	0xFE	
0x1012	0xED	
0x1011	0xBA	
0x1010	0xBE	

Example

What is in $t0 after the following instructions? Assume little endian memory storage.

```
1: ori  $t0  $zero 0xA5C11000
2: addi $t1  $zero 0x10000
3: sw   $t0 ($t1)
```

```
4: lb   $t0 1($t1)
5: sh   $t0 2($t1)
6: lw   $t0  ($t1)
```

ori $t0 $zero 0xA5C11000

$t0:
$t1:

ADDRESS	CONTENTS

lb $t0 1($t1)

$t0:
$t1:

ADDRESS	CONTENTS

addi $t1 $zero 0x10010000

$t0:
$t1:

ADDRESS	CONTENTS

sh $t0 2($t1)

$t0:
$t1:

ADDRESS	CONTENTS

sw $t0 ($t1)

$t0:
$t1:

ADDRESS	CONTENTS

lw $t0 ($t1)

$t0:
$t1:

ADDRESS	CONTENTS

Example

The following program is executed. What is the state of memory and registers after each instruction? If unknown, write '?' Assume little endian memory storage.

```
1: li   $t1  0x10010004
2: li   $t0  0xC0FFEEEE
3: sw   $t0  ($t1)
```

```
4: lb   $t0  3($t1)
5: sh   $t0  2($t1)
6: lw   $t0   ($t1)
```

li $t1 0x10010004

$t0:
$t1:

ADDRESS	CONTENTS

lb $t0 3($t1)

$t0:
$t1:

ADDRESS	CONTENTS

li $t0 0xC0FFEEEE

$t0:
$t1:

ADDRESS	CONTENTS

sh $t0 2($t1)

$t0:
$t1:

ADDRESS	CONTENTS

sw $t0 ($t1)

$t0:
$t1:

ADDRESS	CONTENTS

lw $t0 ($t1)

$t0:
$t1:

ADDRESS	CONTENTS

PC-Relative

Used in branch instructions.

Branch target address: _____

_____

_____

Offset: _____

_____

The branch instruction address and branch target address will always be

_____

So, the offset will always be _____

e.g. _____

_____

The least significant 2 bits of the offset _____

So, _____

_____

_____

Instruction Format

Example: Encode BNE $t0 $t1 LOOP

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x24080000	ADDIU $8,$0,0x00000000	3: ADDIU $t0 $zero 0
	0x00400004	0x24090002	ADDIU $9,$0,0x00000002	4: ADDIU $t1 $zero 2
	0x00400008	0x00000000	NOP	6: LOOP: NOP
	0x0040000c	0x25080001	ADDIU $8,$8,0x00000001	7: ADDIU $t0 $t0 1
	0x00400010	0x1509fffd	BNE $8,$9,0xfffffffd	8: BNE $t0 $t1 LOOP
	0x00400014	0x00000000	NOP	10: NOP

Instruction Format

Opcode for BNE: Rs: Rt:

Determine offset

Branch instruction address:

Branch target address (BTA):

Calculate offset:

To encode in instruction, remove last 2 bits for immediate value:

Pseudo Direct

Used in jump instructions

opcode	target

jump target address = _____