In the following instruction, what method of indexing is used, what will `R0` *contain and what will happen to* `R1`*?*

```
LDR R0, [R1]
```

1

In the following instruction, what method of indexing is used, what will `R0` *contain and what will happen to* `R1`*?*

```
LDR R0, [R1, #4]
```

2

In the following instruction, what method of indexing is used, what will `R0` *contain and what will happen to* `R1`*?*

```
LDR R0, [R1, #4]!
```

3

In the following instruction, what method of indexing is used, what will `R0` *contain and what will happen to* `R1`*?*

```
LDR R0, [R1], #4
```

4

In the following instruction, what method of indexing is used, what will `R0` *contain and what will happen to* `R1` *and* `R2`*?*

```
LDR R0, [R1, R2]
```

5

In the following instruction, what method of indexing is used, what will `R0` *contain and what will happen to* `R1` *and* `R2`*?*

```
LDR R0, [R1, R2, LSL, #2]
```

6

This is called **pre-indexed addressing**.

The value loaded into `R0` *will be the 32 bits stored at the memory address that is equal to the value in* `R1 + 4`.

`R1` *won't be altered at all.*

2

This is called **register-indirect addressing**.

The value loaded into `R0` *will be the 32 bits stored at the memory address that is equal to the value in* `R1`.

`R1` *won't be altered at all.*

1

This is called **post-indexed autoindexed addressing**.

The value loaded into `R0` *will be the 32 bits stored at the memory address that is equal to the value in* `R1 + 4`.

`R1` *will be incremented by* `4` *after the load operation.*

4

This is called **pre-indexed autoindexed addressing**.

The value loaded into `R0` *will be the 32 bits stored at the memory address that is equal to the value in* `R1 + 4`.

`R1` *will be incremented by* `4` *before the load operation.*

3

This is called **scaled register-indexed addressing**.

The value loaded into `R0` *will be the 32 bits stored at the memory address that is equal to the value in* `R1 + (R2 * 4)`.

`R1` *and* `R2` *will stay the same.*

6

This is called **register-indexed addressing**.

The value loaded into `R0` *will be the 32 bits stored at the memory address that is equal to the value in* `R1 + R2`.

`R1` *and* `R2` *will stay the same.*

5