[Minimum operations to Make the integer zero.] 2749

Exp:→ num1 = 3 num = 2

o/p = 3

ops 1 → i=2 ⇒ num1 − 2^2 + (−2)
= 3 − 2 ⇒ 1

ops 2 → i=2 ⇒ num1 − (2^2 + (−2))
= 1 − (2) ⇒ −1

ops = 3 → i=0 ⇒ num1 − (2^0 + (−2))
⇒ −1 + 1 = 0

o/p = 3

num1 = num1 − (2^i + num2)
i = [0, 60]

ops 1 = $num1_1$ = (num1) − (2^{i1} + num2)

ops 2 = $num1_2$ = ($num1_1$) − (2^{i2} + num2)

$num1_2$ = [num1 − (2^{i1} + num2)] − (2^{i2} + num2)

ops 3 ⇒ $num1_3$ = ($num1_2$) − (2^{i3} + num2)

$num1_3$ = [num1 − (2^{i1} + num2)] − (2^{i2} + num2) − (2^{i3} + num2)

$num1_3$ =
.
.
.
.

*th ops 0 =

num1 − t
↓
val

So ques
for give
operatio
num
L.H.S

num 1
let t=1
3 −

t=2
3 −

$num_3 = num1 - (2^{i1} + 2^{i2} + 2^{i3}) - 3 * num2$

⋮

t^{th} ops $0 = num1 - (2^{i1} + 2^{i2} + 2^{i3} \cdots 2^{it}) - (t * num2)$

$num1 - t * num2 = 2^{i1} + 2^{i2} + 2^{i3} \cdots 2^{it}$

↓

val = express in ~~t powers~~ sum of t powers of 2.

So question becomes →
for given num1 & num2 and given operation count t →
num1 − t*num2 = express in sum of t power of 2.
L.H.S R.H.S

num1 = 3 num2 = −2

let t = 1

$3 - 1 * 2 = 5$
L.H.S

Binary
$101 = 2^2 + 2^0$
t = 2
while my t was 1

t = 2

$3 - 2 * 2 = 7$

$111 = 2^2 + 2^1 + 2^0$
t = 3
Need 3 2' powers

while my t = 2 ≠ 3

Val 9 = 1001 = $2^0+2^0+2^0+2^0+2^0+2^0+2^0+2^0+2^0$

9 times = max * value = 9 ⇒ Val

Let t = 3

3 − 3*(−2)

= 9 ⇒ 1001

2^3+2^0 ⇒ t = 2

Minimum need two 2 power of t.

or

$= 2^2+2^2+2^0 = 9$ Valid

t = 3 min

valid

$2^2+2^0+2^0+2^0+2^0+2^0$

t = 6

val = num1 − t * num2

min_bits = __builtin_popcount(val);

min_bits <= t <= val

So start t from 1 & check when condition is being satisfied.

```
t = 1
while (true)
{
    val = num1 - t*num2;
    if (__builtin_popcount(val) <= t
        <= val)
        return t;
    t++;
}
```

if (val < 0) return -1;

num1 = 5 num2 = 7

t = 1 val = 5 - 1 * 7 = -2

so invalid return -1

Constraints

num1 = 10^9

min num2 = 10^9

val = num1 - t * num2

= 10^9 - t (-10^9)

val = 10^9 (1 + t)

t power of 2

number <= 2^t

number = 6 <= 2^3 (t = 3)

1 1 0

(2 bits)

$2^2 + 2^1$

"Any number <= 2^t can be represented in min. t bits

val <= 2^t

10^9 (1 + t) <= 2^t

t ≈ 35

So t = 1

while (true) ✗ while (t = 36 max)

```
for (t = 1; t <= 36; i++)
{
  val = num1 - t * num2;
  if (val < 0) return -1;
  if (bits <= t <= val)
    { return t;
    }
}
```

Approach-1

```
int solve(num1, num2)
{ int t=0
  while(true)
  {
  long long val = (long long)num1 - (long long)t * num2;
  if (val < 0)
      return -1;
  if (__builtin_popcount(val) <= t && t <= val)
    { return t;
    }
  t++;
}
return -1;
```

__builtin

Approach-2

```
{
  for
  {
  long
  if(
  if(
    {
    }
  }
  retur
}
```

--built in popcount --> (log)

Approach-2

```
int solve( ){
    for(int t=1; t<=36; t++)
    {
        long long val = (long long) num1 *
                        (long long) num2;
        if(val < 0) return -1;
        if(__builtin_popcountll(val) <= t &&
                t <= val)
        {
            return t;
        }
    }
    return -1;
}
```