Let \( n \) and \( r \) be natural variables in the refinement

\[
P \iff \begin{cases} n=1 & \text{then } r:=0 \\
\text{else } n:= \text{div} \ n \ 2. & \text{P. } r:= r+1 \end{cases}
\]

Suppose the operations \text{div} and + each take time 1 and all else is free (even the call is free). Insert appropriate time increments, and find an appropriate \( P \) to express the execution time in terms of

(a) the initial values of the memory variables. Prove the refinement for your choice of \( P \).

§ With time increments added, I must prove

\[
P \iff \begin{cases} n=1 & \text{then } r:=0 \\
\text{else } t:= t+1, \ n:= \text{div} \ n \ 2. & \text{P. } t:= t+1, \ r:= r+1 \end{cases}
\]

How should we choose \( P \)? Execution of \( P \) proceeds as follows. If \( n \) is initially 0, then \( n \) is divided by 2, making it again 0, and we are in an infinite loop. If \( n \) is initially positive, then it is repeatedly divided by 2 (rounding down) until it becomes 1, then \( r \) is assigned 0, then \( r \) is incremented as many times as \( n \) was divided by 2. The number of times \( n \) is divided by 2 until it becomes 1 is the logarithm (base 2) of \( n \). This may not be obvious, so I can easily code this procedure in any implemented programming language I like, and run it for a variety of initial values for \( n \) and \( r \) and for initial time 0, and see that the final value of \( t \) is \( 2 \times \text{floor} \ (\text{log} \ n) \). So \( P \) can be

\[(n=0 \implies t'=\infty) \land (n>0 \implies t'= t+2 \times \text{floor} \ (\text{log} \ n))\]

But \text{floor} is an awkward function to work with, so I'll get rid of it by replacing the exact context, and

\[P \implies t' \leq t+2 \times \text{log} \ n\]

I prove it in parts (each conjunct separately), and I prove each part by cases.

First part, first case:

\[
\begin{align*}
(n=0 & \implies t'=\infty) \iff n=1 \land (r:=0) \quad \text{portation} \\
\equiv n=0 \land n=1 \land (r:=0) & \implies t'=\infty \quad \text{context: } n=0 \\
\equiv \bot \land (r:=0) & \implies t'=\infty \quad \text{context: } n=0 \\
\equiv \bot & \implies t'=\infty \quad \text{context: } n=0 \\
\equiv \top &
\end{align*}
\]

First part, last case:

\[
\begin{align*}
(n=0 & \implies t'=\infty) \iff n+1 \land (t:= t+1, \ n:= \text{div} \ n \ 2. \ n=0 \implies t'=\infty. \ t:= t+1, \ r:= r+1) \quad \text{portation and expand final assignment} \\
\equiv n=0 \land n+1 \land (t:= t+1, \ n:= \text{div} \ n \ 2. \ n=0 \implies t'=\infty. \ t:= t+1, \ r'=r+1 \land n'=n \land t'=t+1) & \implies t'=\infty \quad \text{simplify, and substitution law in two parts} \\
\equiv n=0 \land (\text{div} \ n \ 2 = 0 \implies t'=\infty. \ r'=r+1 \land n'=n \land t'=t+1) & \implies t'=\infty \quad \text{eliminate dependent composition} \\
\equiv n=0 \land (\exists r'', n'', t''). \ (\text{div} \ n \ 2 = 0 \implies t''=\infty) \land r''=r'+1 \land n''=n' \land t''=t''+1) & \implies t'=\infty \quad \text{one-point} \\
\equiv n=0 \land (n' \land t'=t') & \implies t'=\infty \quad \text{absorption and specialization} \\
\equiv \top &
\end{align*}
\]

Last part, first case:

\[
(n>0 \implies t' \leq t+2 \times \text{log} \ n) \iff n=1 \land (r:=0) \quad \text{portation and expand assignment} \\
\equiv n=1 \land r'=0 \land n'=n \land t'=t \quad \text{context, and } \text{log} \ 1 = 0 \\
\equiv \top
\]

Last part, last case:
(n>0 ⇒ t' ≤ t + 2 × log n)

⇐ n>1 ∧ (t:=t+1. n:= div n 2. n>0 ⇒ t' ≤ t + 2 × log n) portation and expand final assignment

⇐ t' ≤ t + 2 × log n

⇐ n>1 ∧ (t:=t+1. n:= div n 2. n>0 ⇒ t' ≤ t + 2 × log n) substitution law in two parts

⇐ t' ≤ t + 2 × log n

⇐ n>1 ∧ (t:=t+1. n:= div n 2. n>0 ⇒ t' ≤ t + 2 × log n) eliminate dependent composition

⇐ t' ≤ t + 2 × log n

⇐ n>1 ∧ (∃r'', n'', t'': (div n 2>0 ⇒ t'' ≤ t+1+ 2 × log(div n 2)) one-point for n'' and t''

⇐ t' ≤ t + 2 × log n

⇐ n>1 ∧ (∃r'': (div n 2>0 ⇒ t'' ≤ t+2+ 2 × log(div n 2)) ∧ r''=r''+1) distributive

⇐ t' ≤ t + 2 × log n

⇐ n>1 ∧ (∃r'': nat: r''=r''+1) ∧ (div n 2>0 ⇒ t'' ≤ t+2+ 2 × log(div n 2)) in preparation for one-point, rewrite r''=r''+1 and make r'': nat an explicit conjunct

⇐ t' ≤ t + 2 × log n

⇐ n>1 ∧ (∃r'': nat: r''=r''+1 ∧ (r'':nat)) ∧ (div n 2>0 ⇒ t' ≤ t+2+ 2 × log(div n 2)) now use one-point

⇐ t' ≤ t + 2 × log n

⇐ n>1 ∧ (r'−1: nat) ∧ (div n 2>0 ⇒ t' ≤ t+2+ 2 × log(div n 2)) simplify div n 2>0

⇐ t' ≤ t + 2 × log n

⇐ n>1 ∧ r'≥1 ∧ (n>1 ⇒ t' ≤ t + 2 + 2 × log(div n 2)) increase div n 2 to n/2 this will increase t + 2 + 2 × log(div n 2) this will weaken t' ≤ t + 2 + 2 × log(div n 2)

this will weaken n>1 ⇒ t' ≤ t + 2 + 2 × log(div n 2) this will weaken n>1 ⇒ n' ≥ 1 ∧ (n>1 ⇒ t' ≤ t + 2 + 2 × log(div n 2)) this will strengthen t' ≤ t + 2 + 2 × log n ¬ n>1 ∧ r'≥1 ∧ (n>1 ⇒ t' ≤ t + 2 + 2 × log(div n 2)) so we need to put ¬ in the left margin

⇐ t' ≤ t + 2 × log n ¬ n>1 ∧ r'≥1 ∧ (n>1 ⇒ t' ≤ t + 2 + 2 × log(n/2)) drop r'≥1 this will weaken n>1 ∧ r'≥1 ∧ (n>1 ⇒ t' ≤ t + 2 + 2 × log(n/2)) this will strengthen t' ≤ t + 2 × log n ¬ n>1 ∧ r'≥1 ∧ (n>1 ⇒ t' ≤ t + 2 + 2 × log(n/2)) so again we need to put ¬ in the left margin

⇐ t' ≤ t + 2 × log n ¬ (n>1 ⇒ t' ≤ t + 2 + 2 × log(n/2)) discharge and simplify

⇐ t' ≤ t + 2 × log n ¬ n>1 ∧ t' ≤ t + 2 × log n specialization

⇐ T

(b) the final values of the memory variables. Prove the refinement for your choice of \( P \).

§ I prove

\[
\begin{align*}
    t &= \text{t+2}\times r' \\
    \text{if } n=1 &\text{ then } r:=0 \\
    \text{else } t := t+1. n := \text{div n 2}. t' &= \text{t+2}\times r'. t := t+1. r := r+1 \end{align*}
\]
by cases. First case:

\[
\begin{align*}
    t &= \text{t+2}\times r' \\
    \text{if } n=1 &\text{ then } r := 0 \\
    \text{else } t := t+1. n := \text{div n 2}. t' &= \text{t+2}\times r'. t := t+1. r := r+1 \end{align*}
\]
expand assignment

\[
\begin{align*}
    t &= \text{t+2}\times r' \\
    \text{if } n=1 &\text{ then } r := 0 \\
    \text{else } t := t+1. n := \text{div n 2}. t' &= \text{t+2}\times r'. t := t+1. r := r+1 \end{align*}
\]

\( T \)
Last case:

\[ t' = t + 2 \times r' \quad \iff \quad n \neq 1 \land (t := t + 1. \ n := \text{div} \ n \ 2. \ t' = t + 2 \times r'. \ t := t + 1. \ r := r + 1) \]

expand final assignment

\[ t' = t + 2 \times r' \quad \iff \quad n \neq 1 \land (t := t + 1. \ n := \text{div} \ n \ 2. \ t' = t + 2 \times r'. \ t := t + 1. \ r := r + 1 \land n' = n \land t' = t) \]

substitution law in two parts

\[ t' = t + 2 \times r' \quad \iff \quad n \neq 1 \land (t' = t + 1 + 2 \times r'. \ r' = r + 1 \land n' = n \land t' = t + 1) \]

one-point

\[ t' = t + 2 \times r' \quad \iff \quad n \neq 1 \land t' = t + 2 + 2 \times (r' - 1) \]

\[ \top \]