CFG

Fibonacci

Text

Description automatically generated

La iR in immagine, rappresenta, la dichiarazione e definizione della funzione printf del modulo Fibonacci.

Come è possibile notare, la IR di questa funzione non contiene nessun Basic Block.

Text

Description automatically generated

Entry -> Basic Block 2 Fallthrough

Basic Block 2 -> Basic Block 11 True Se il registro %4 è uguale a 0

Basic Block 2 -> Basic Block 12 True Se il registro %4 è uguale a 1

Basic Block 2 -> Basic Block 5 False Altrimenti

Basic Block 5 -> Basic Block 2 True

Basic Block 11 -> Basic Block 12 Fallthrough

La rappresentazione intermedia di LLVM vede le chiamate a funzione come singole istruzioni che non alterano il controllo di flusso, per quanto riguarda la definizione di Basic Block. Pertanto LLVM ritiene non necessario creare ulteriori Basic Block in presenza di chiamate in funzione, in quanto l’istruzione stessa contiene al suo interno questa informazione.

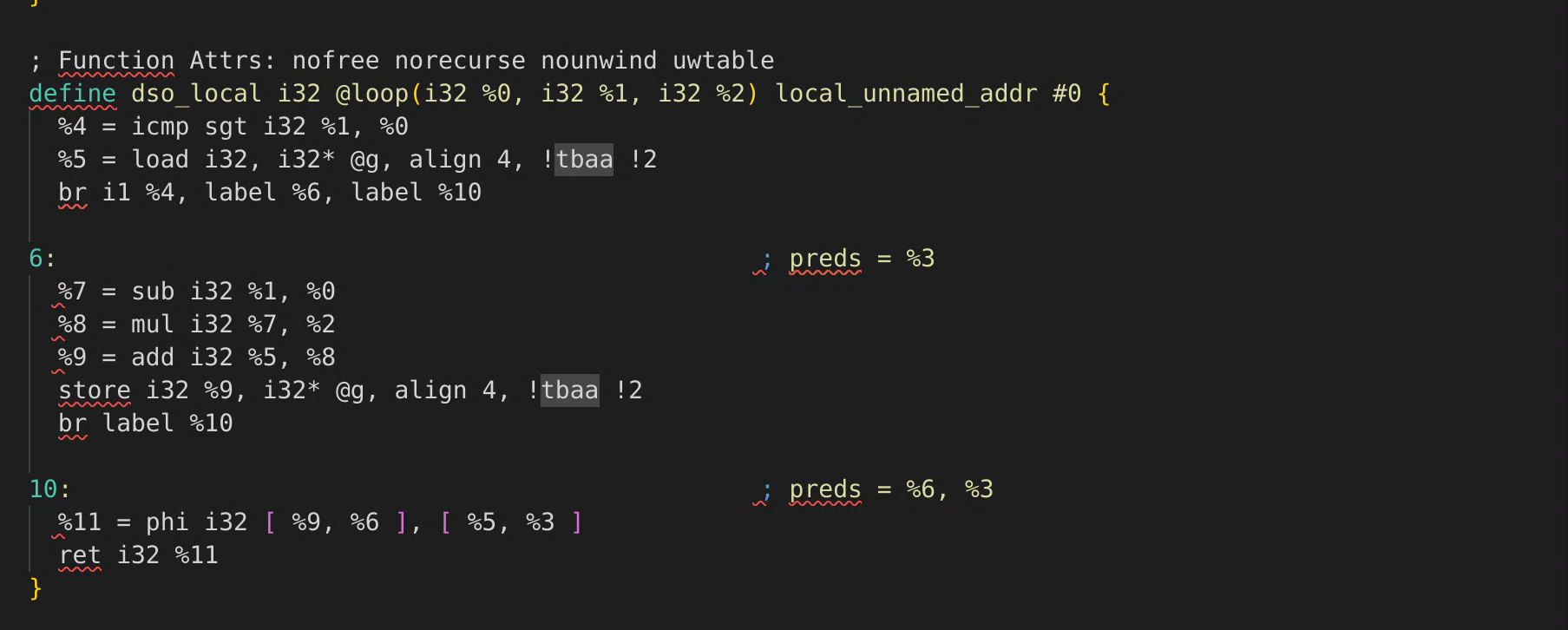
Loop

Text

Description automatically generated

La iR in immagine, rappresenta, la dichiarazione e definizione della funzione g\_incr del modulo Loop.

Come è possibile notare, la IR di questa funzione non contiene nessun Basic Block.



Entry -> Basic Block 6 True

Entry -> Basic Block 10 False

Basic Block 6 -> Basic Block 10 Fallthrough

Si riporta a sinistra il bytecode prodotto in modalità -O0, ovvero senza ottimizzazioni, e a destra la corrispondente versione in modalità -O2

Immagine che contiene testo

Descrizione generata automaticamenteLoop:

Immagine che contiene testo

Descrizione generata automaticamente

Si può notare innanzitutto che tutte le istruzioni alloca vengono eliminate, in quanto non necessarie, dal pass di ottimizzazione “mem2reg”.

Inoltre, nella funzione loop, viene ridotto il numero di BasicBlock. Il loop presente all’interno della funzione loop viene eliminato sostituibile con 3 semplici istruzioni: una sottrazione, una moltiplicazione, e un’addizione. Questo è stato possibile solamente in seguito all’inlining della funzione g\_incr.

Eseguendo il seguente comando:  
clang -O2 -emit-llvm -S -c test/Loop.c -o test/LoopO2.ll -Rpass=.\*  
vengono stampate in dettaglio le ottimizzazioni che vengono effettuate.

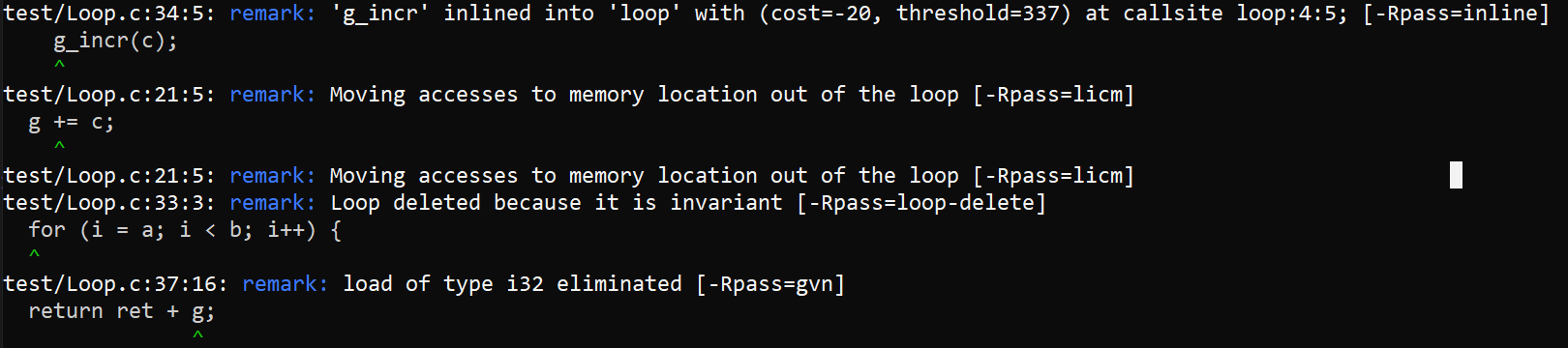


Immagine che contiene testo

Descrizione generata automaticamenteImmagine che contiene testo

Descrizione generata automaticamenteFibonacci:

In questo caso un’ottimizzazione di fondamentale importanza è l’eliminazione di una chiamata ricorsiva, che porta a miglioramenti estremi dei tempi di esecuzione. Una delle due chiamate ricorsive (precisamente Fibonacci(n-2)) viene eliminata grazie all’utilizzo di un loop. Si può notare infine che il numero di BasicBlock è diminuito grazie all’utilizzo di istruzioni phi.

Ancora una volta, con l’opzione -Rpass=.\* si possono visualizzare le varie ottimizzazioni.

