

CFG LOOP. C CON COMPILAZIONE - O2

CFG Funzione Loop

```
; Function Attrs: nofree norecurse nosync nounwind uwtable
define dso_local i32 @loop(i32 noundef %0, i32 noundef %1, i32
noundef %2) local_unnamed_addr #1 {
    %4 = load i32, i32* @g, align 4, !tbaa !3
    %5 = icmp sgt i32 %1, %0
    br i1 %5, label %6, label %10
```

LLVM IR

BB1

```
6:                                ; preds = %3
    %7 = sub i32 %1, %0
    %8 = mul i32 %7, %2
    %9 = add i32 %4, %8
    store i32 %9, i32* @g, align 4, !tbaa !3
    br label %10
```

BB2

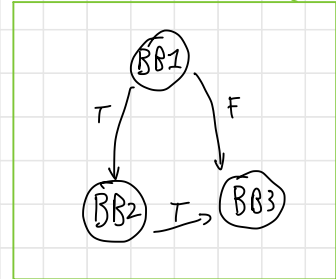
```
10:                               ; preds = %6, %3
    %11 = phi i32 [ %9, %6 ], [ %4, %3 ]
    ret i32 %11
}
```

BB3

```
int loop(int a, int b, int c) {
    int i, ret = 0;
    for (i = a; i < b; i++) {
        g_incr(c);
    }
}
```

SORGENTE

CFG Funzione Loop



CFG g_increment

```
int g;

int g_incr(int c) {
    g += c;
    return g;
}
```

SORGENTE

```
@g = dso_local local_unnamed_addr global i32 0, align 4
```

```
define dso_local i32 @g_incr(i32 noundef %0) local_unnamed_addr #0
{
    %2 = load i32, i32* @g, align 4, !tbaa !3
    %3 = add nsw i32 %2, %0
    store i32 %3, i32* @g, align 4, !tbaa !3
    ret i32 %3
}
```

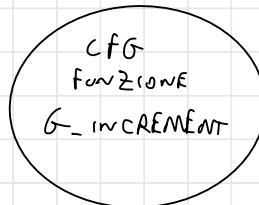
LLVM IR

BB

CFG Funzione g_increment



CFG PROGRAMMA

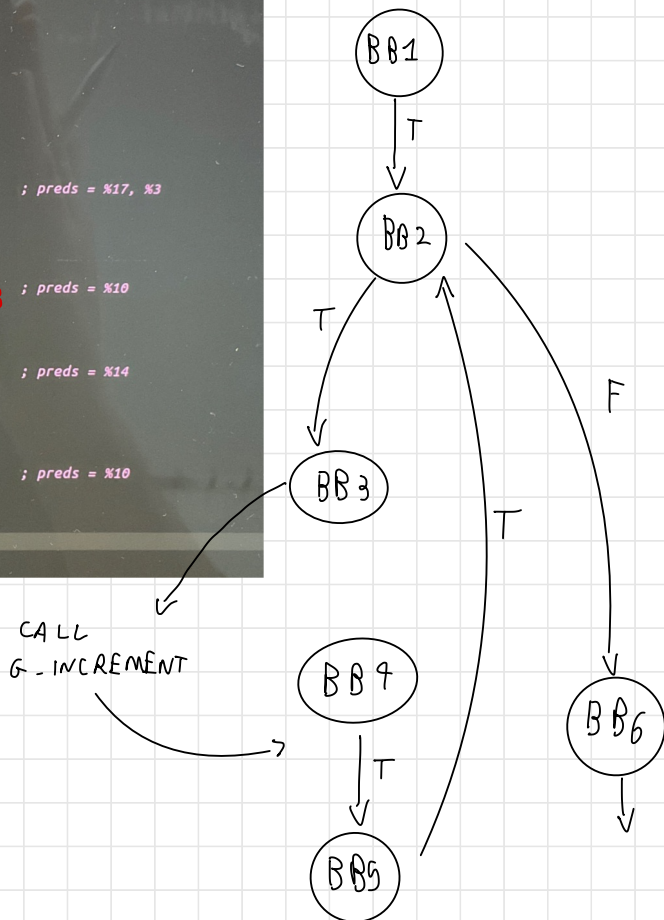


In questo caso, i due CFG relativi alle singole funzioni non sono collegati in quanto, compilando il codice sorgente usando l'ottimizzazione -O2, LLVM ha eseguito una ottimizzazione per cui il corpo della funzione g_increment e' stato direttamente inserito all'interno del basic block relativo al corpo del loop della funzione Loop.

CFG Loop.c con compilazione - O8

CFG FUNZIONE LOOP

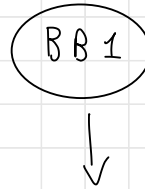
```
20 ; Function Attrs: noline nounwind optnone uwtable
21 define dso_local @loop(i32 @loop(i32 noundef %0, i32 noundef %1, i32 noundef %2) #0 {
22     %4 = alloca i32, align 4 ;a
23     %5 = alloca i32, align 4 ;b
24     %6 = alloca i32, align 4 ;c
25     %7 = alloca i32, align 4 ;t
26     %8 = alloca i32, align 4 ;ret
27     store i32 %0, i32* %4, align 4
28     store i32 %1, i32* %5, align 4
29     store i32 %2, i32* %6, align 4
30     store i32 0, i32* %8, align 4
31     %9 = load i32, i32* %4, align 4
32     store i32 %9, i32* %7, align 4
33     br label %10
34
35 10:
36     %11 = load i32, i32* %7, align 4 ; preds = %17, %3
37     %12 = load i32, i32* %5, align 4
38     %13 = icmp slt i32 %11, %12
39     br i1 %13, label %14, label %20
40
41 14:
42     %15 = load i32, i32* %6, align 4 ; preds = %10
43     %16 = call @g_incr(i32 noundef %15)
44     br label %17
45
46 17:
47     %18 = load i32, i32* %7, align 4 ; preds = %14
48     %19 = add nsw i32 %18, 1
49     store i32 %19, i32* %7, align 4
50     br label %10, !llvm.loop !10
51
52 20:
53     %21 = load i32, i32* %8, align 4 ; preds = %10
54     %22 = load i32, i32* @g, align 4
55     %23 = add nsw i32 %21, %22
56     ret i32 %23
57 }
```



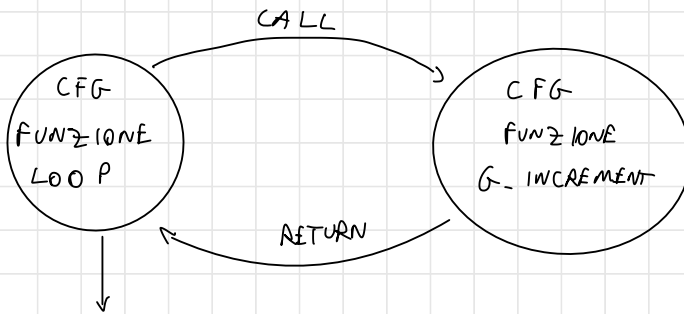
CFG FUNZIONE G_INCREMENT

```
7  
8 ; Function Attrs: noinline nounwind optnone uwtable  
9 define dso local @g_incr(i32@noundef %0) #0 {  
10     %2 = alloca i32, align 4  
11     store i32 %0, i32* %2, align 4  
12     %3 = load i32, i32* %2, align 4  
13     %4 = load i32, i32* @g, align 4  
14     %5 = add nsw i32 %4, %3  
15     store i32 %5, i32* @g, align 4  
16     %6 = load i32, i32* @g, align 4  
17     ret i32 %6  
18 }
```

BB 1



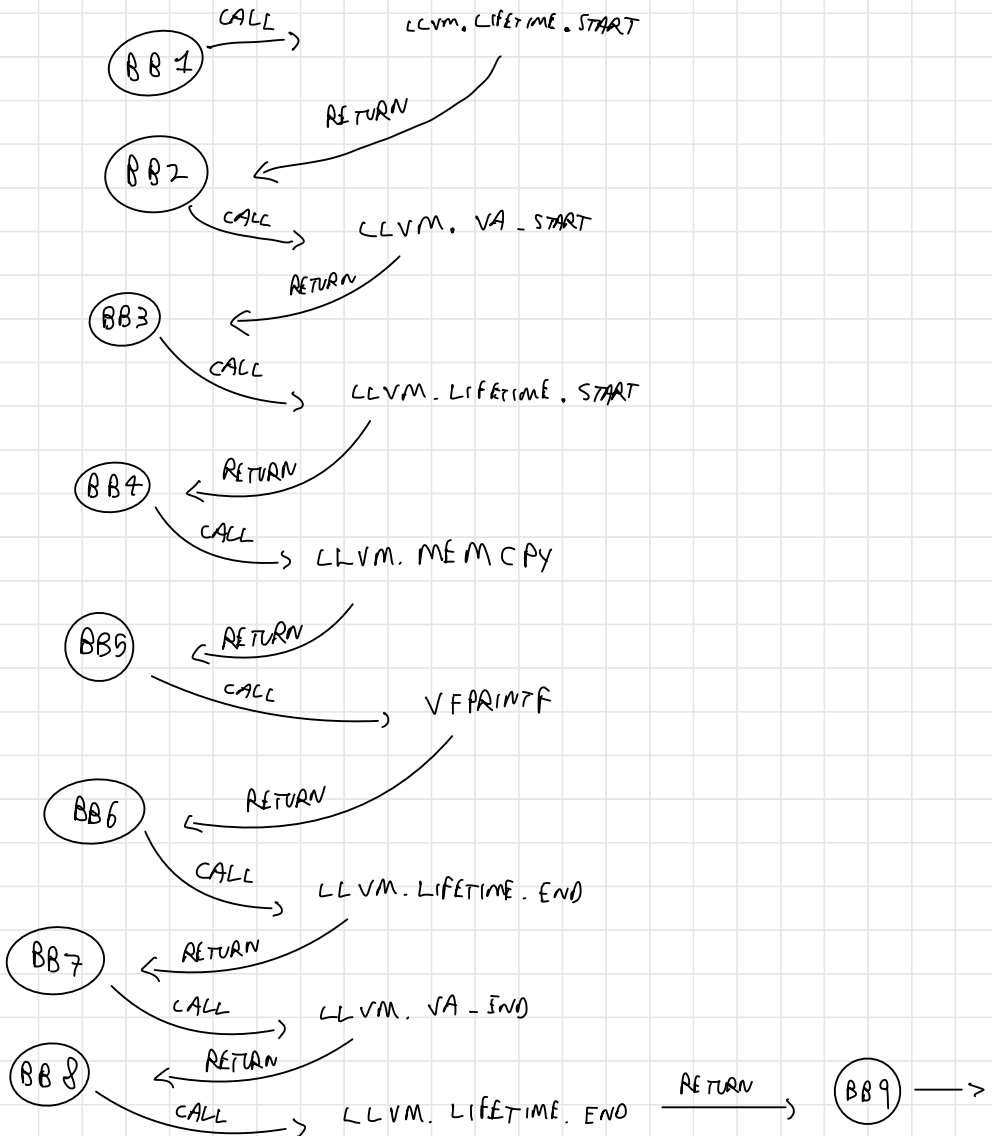
CFG INTERO PROGRAMMA



FILE FIBONACCI. C CON COMPILAZIONE - 02

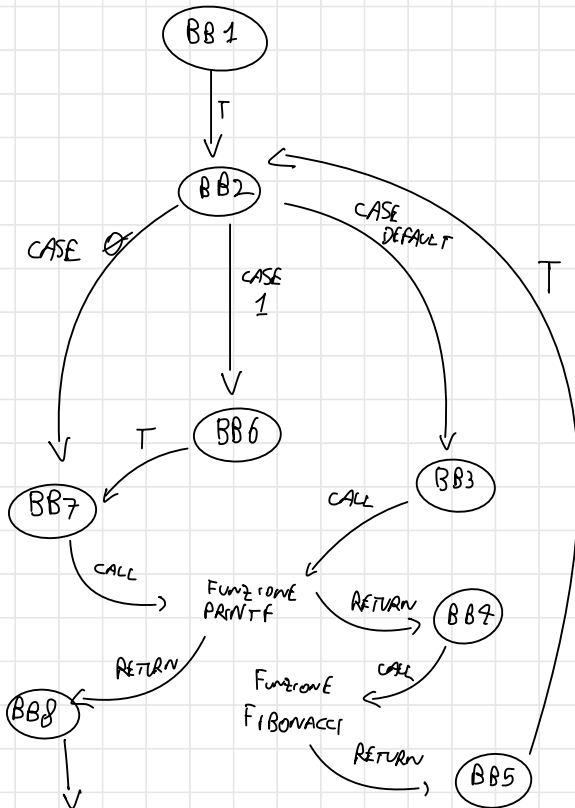
CFG FUNZIONE PRINTF

```
; Function Attrs: nofree nounwind uwtable
define dso_local @printf@18@nocapture noinline readonly @0, ... local_unnamed_addr @0 {
  %2 = alloca %"struct.std::__va_list", align 8
  %3 = alloca %"struct.std::__va_list", align 8
  %4 = bitcast %"struct.std::__va_list" %2 to i8*
  call void @llvm.lifetime.start.p0i8(i64 32, i8* nonnull %4) #5
  call void @llvm.va_start(i8* nonnull %4)
  %5 = load %struct._IO_FILE*, %struct._IO_FILE** @stdout, align 6, i8as i10
  %6 = bitcast %"struct.std::__va_list" %3 to i8*
  call void @llvm.lifetime.start.p0i8(i64 32, i8* nonnull %6) #5
  call void @llvm.memcpy.p0i8.p0i8.i64(i8* nonnull align 8 dereferenceable(32) %6, i8* nonnull align 8 dereferenceable(32) %4, i64
  32, i8 false), i8as struct i14
  %7 = call i32 @vfprintf(%struct._IO_FILE* nonnull %5, i8* nonnull %0, %"struct.std::__va_list" nonnull %3)
  call void @llvm.lifetime.end.p0i8(i64 32, i8* nonnull %6) #5
  call void @llvm.va_end(i8* nonnull %4)
  call void @llvm.lifetime.end.p0i8(i64 32, i8* nonnull %4) #5
  ret i32 %7
}
```



```

4: define dso local i32 @Fibonacci(i32 noundef #0) local_unnamed_addr #0 {
5:   br label %2
6:
7:
8:   %3 = phi i32 [ 0, %1 ], [ %10, %5 ]
9:   %4 = phi i32 [ %0, %1 ], [ %7, %5 ]
10:  switch i32 %4, label %5 [
11:    i32 0, label %12
12:    i32 1, label %11
13:  ]
14:
15:
16:
17:
18:
19:
20:
21:
22:
23:
24:
25:
26:
27:
28:
29:
30:
31:
32:
33:
34:
35:
36:
37:
38:
39:
40:
41:
42:
43:
44:
45:
46:
47:
48:
49:
50:
51:
52:
53:
54:
55:
56:
57:
58:
59:
60:
61:
62:
63:
64:
65:
66:
67:
68:
69:
70:
71:
72:
73:
74:
75:
76:
77:
78:
79:
80:
81:
82:
83:
84:
85:
86:
87:
88:
89:
90:
91:
92:
93:
94:
95:
96:
97:
98:
99:
100:
101:
102:
103:
104:
105:
106:
107:
108:
109:
110:
111:
112:
113:
114:
115:
116:
117:
118:
119:
120:
121:
122:
123:
124:
125:
126:
127:
128:
129:
130:
131:
132:
133:
134:
135:
136:
137:
138:
139:
140:
141:
142:
143:
144:
145:
146:
147:
148:
149:
150:
151:
152:
153:
154:
155:
156:
157:
158:
159:
160:
161:
162:
163:
164:
165:
166:
167:
168:
169:
170:
171:
172:
173:
174:
175:
176:
177:
178:
179:
180:
181:
182:
183:
184:
185:
186:
187:
188:
189:
190:
191:
192:
193:
194:
195:
196:
197:
198:
199:
200:
201:
202:
203:
204:
205:
206:
207:
208:
209:
210:
211:
212:
213:
214:
215:
216:
217:
218:
219:
220:
221:
222:
223:
224:
225:
226:
227:
228:
229:
230:
231:
232:
233:
234:
235:
236:
237:
238:
239:
240:
241:
242:
243:
244:
245:
246:
247:
248:
249:
250:
251:
252:
253:
254:
255:
256:
257:
258:
259:
260:
261:
262:
263:
264:
265:
266:
267:
268:
269:
270:
271:
272:
273:
274:
275:
276:
277:
278:
279:
280:
281:
282:
283:
284:
285:
286:
287:
288:
289:
290:
291:
292:
293:
294:
295:
296:
297:
298:
299:
300:
301:
302:
303:
304:
305:
306:
307:
308:
309:
310:
311:
312:
313:
314:
315:
316:
317:
318:
319:
320:
321:
322:
323:
324:
325:
326:
327:
328:
329:
330:
331:
332:
333:
334:
335:
336:
337:
338:
339:
340:
341:
342:
343:
344:
345:
346:
347:
348:
349:
350:
351:
352:
353:
354:
355:
356:
357:
358:
359:
360:
361:
362:
363:
364:
365:
366:
367:
368:
369:
370:
371:
372:
373:
374:
375:
376:
377:
378:
379:
380:
381:
382:
383:
384:
385:
386:
387:
388:
389:
390:
391:
392:
393:
394:
395:
396:
397:
398:
399:
400:
401:
402:
403:
404:
405:
406:
407:
408:
409:
410:
411:
412:
413:
414:
415:
416:
417:
418:
419:
420:
421:
422:
423:
424:
425:
426:
427:
428:
429:
430:
431:
432:
433:
434:
435:
436:
437:
438:
439:
440:
441:
442:
443:
444:
445:
446:
447:
448:
449:
450:
451:
452:
453:
454:
455:
456:
457:
458:
459:
460:
461:
462:
463:
464:
465:
466:
467:
468:
469:
470:
471:
472:
473:
474:
475:
476:
477:
478:
479:
480:
481:
482:
483:
484:
485:
486:
487:
488:
489:
490:
491:
492:
493:
494:
495:
496:
497:
498:
499:
500:
501:
502:
503:
504:
505:
506:
507:
508:
509:
510:
511:
512:
513:
514:
515:
516:
517:
518:
519:
520:
521:
522:
523:
524:
525:
526:
527:
528:
529:
530:
531:
532:
533:
534:
535:
536:
537:
538:
539:
540:
541:
542:
543:
544:
545:
546:
547:
548:
549:
550:
551:
552:
553:
554:
555:
556:
557:
558:
559:
560:
561:
562:
563:
564:
565:
566:
567:
568:
569:
570:
571:
572:
573:
574:
575:
576:
577:
578:
579:
580:
581:
582:
583:
584:
585:
586:
587:
588:
589:
590:
591:
592:
593:
594:
595:
596:
597:
598:
599:
600:
601:
602:
603:
604:
605:
606:
607:
608:
609:
610:
611:
612:
613:
614:
615:
616:
617:
618:
619:
620:
621:
622:
623:
624:
625:
626:
627:
628:
629:
630:
631:
632:
633:
634:
635:
636:
637:
638:
639:
640:
641:
642:
643:
644:
645:
646:
647:
648:
649:
650:
651:
652:
653:
654:
655:
656:
657:
658:
659:
660:
661:
662:
663:
664:
665:
666:
667:
668:
669:
670:
671:
672:
673:
674:
675:
676:
677:
678:
679:
680:
681:
682:
683:
684:
685:
686:
687:
688:
689:
690:
691:
692:
693:
694:
695:
696:
697:
698:
699:
700:
701:
702:
703:
704:
705:
706:
707:
708:
709:
710:
711:
712:
713:
714:
715:
716:
717:
718:
719:
720:
721:
722:
723:
724:
725:
726:
727:
728:
729:
730:
731:
732:
733:
734:
735:
736:
737:
738:
739:
740:
741:
742:
743:
744:
745:
746:
747:
748:
749:
750:
751:
752:
753:
754:
755:
756:
757:
758:
759:
760:
761:
762:
763:
764:
765:
766:
767:
768:
769:
770:
771:
772:
773:
774:
775:
776:
777:
778:
779:
780:
781:
782:
783:
784:
785:
786:
787:
788:
789:
790:
791:
792:
793:
794:
795:
796:
797:
798:
799:
800:
801:
802:
803:
804:
805:
806:
807:
808:
809:
810:
811:
812:
813:
814:
815:
816:
817:
818:
819:
82
```



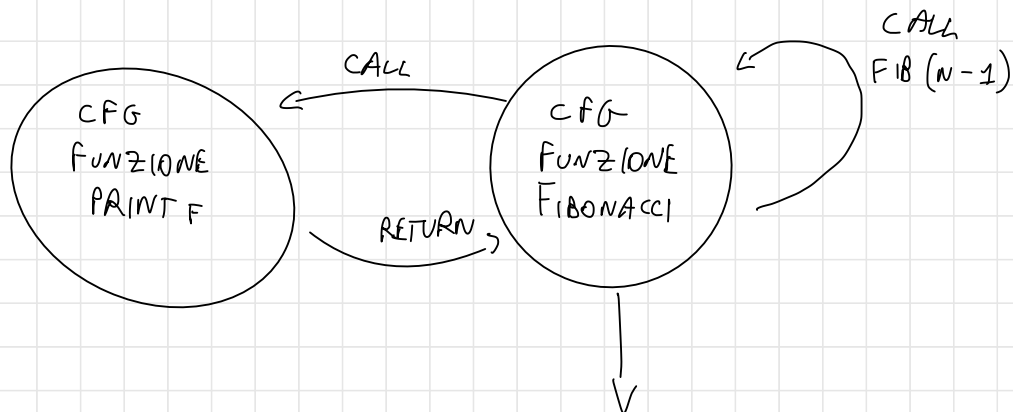
Il codice mostrato nell'immagine presenta già delle grosse ottimizzazioni per permettere di ridurre il numero di chiamate ricorsive da effettuare; infatti, all'interno del registro %10 viene via a via salvato il valore della chiamata della funzione di Fibonacci del valore immediatamente precedente a quello che si sta considerando in questo momento.

In questo modo, per ogni numero maggiore di 1, invece che effettuare due chiamate ricorsive ($Fib(n-1)$ e $Fib(n-2)$), viene chiamato solo $Fib(n-1)$ visto che di sicuro, prima di essere giunto alla chiamata $Fib(n-1)$ devo anche aver effettuato la chiamata per $Fib(n-2)$, e quindi tale valore lo ho già calcolato (e' presente ogni volta all'interno del registro %10).

Un'altra ottimizzazione presente all'interno di questo codice fa' riferimento alla gestione dei due casi base: in pratica, invece che definire due blocchi identici, vengono definiti due basic blocks separati: uno e' effettivamente quello che presenta la stampa del valore, mentre l'altro presenta solamente un salto incondizionato.

In questa maniera, gestendo l'arrivo del flusso di controllo al basic block avente le istruzioni mediante l'istruzione PHI, siamo in grado di capire quale valore dobbiamo andare a stampare, se 0 oppure 1.

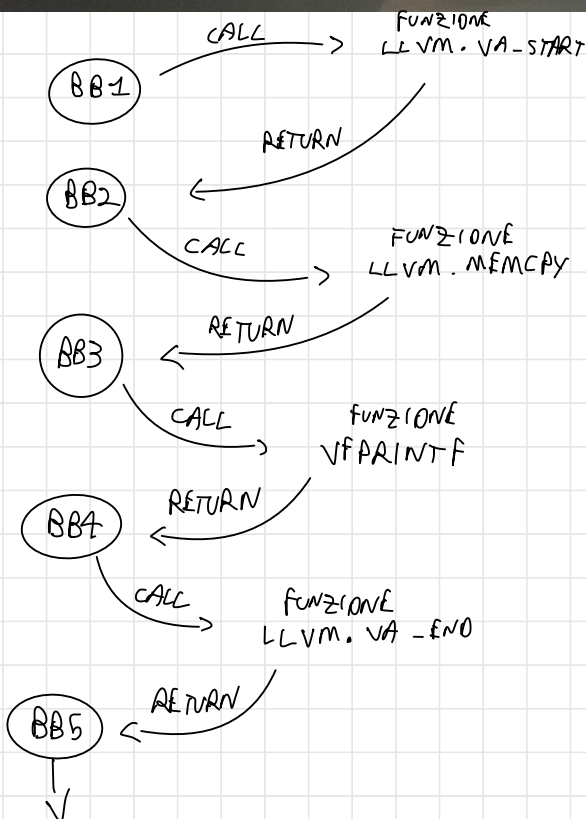
CFG PROGRAMMA FIBONACCI



FILE FIBONACCI.C CON COMPILAZIONE - 00

CFG FUNZIONE PRINTF

```
; Function Attrs: noinline nounwind optnone uwtable
define dso_local @printf(i8* noundef %0, ...) #0 {
    %2 = alloca i8*, align 8
    %3 = alloca i32, align 4
    %4 = alloca %"struct.std::_va_list", align 8
    %5 = alloca %"struct.std::_va_list", align 8
    store i8* %0, i8** %2, align 8
    %6 = bitcast %"struct.std::_va_list"* %4 to i8*
    call void @llvm.va_start(i8* %6)
    %7 = load %struct._IO_FILE*, %struct._IO_FILE** @stdout, align 8
    %8 = load i8*, i8** %2, align 8
    %9 = bitcast %"struct.std::_va_list"* %5 to i8*
    %10 = bitcast %"struct.std::_va_list"* %4 to i8*
    call void @llvm.memcpy.p0i8.p0i8.i64(i8* align 8 %9, i8* align 8 %10, i64 22, i1 false)
    %11 = call i32 @vfprintf(%struct._IO_FILE* noundef %7, i8* noundef %8, %"struct.std::_va_list"* noundef %5)
    store i32 %11, i32* %3, align 4
    %12 = bitcast %"struct.std::_va_list"* %4 to i8*
    call void @llvm.va_end(i8* %12)
    %13 = load i32, i32* %3, align 4
    ret i32 %13
}
```



CFG FUNZIONE FIBONACCI

```

50 ; Function Attrs: noinline nounwind optnone @naked
51 define dso_local @i32 @Fibonacci(i32 noundef %0) #0 {
52     %2 = alloca i32, align 4
53     %3 = alloca i32, align 4
54     store i32 %0, i32* %3, align 4
55     %4 = load i32, i32* %3, align 4
56     %5 = icmp eq i32 %4, 0
57     br i1 %5, label %6, label %8

58
59
60 ; preds = %1
61 %7 = call @i32 (@i32, ...) @printf(i8* noundef getelementptr @inbounds ([9 x i8], [9 x i8]* @.str.1, i64 0, i64 0))
62     store i32 0, i32* %2, align 4
63     br label %27

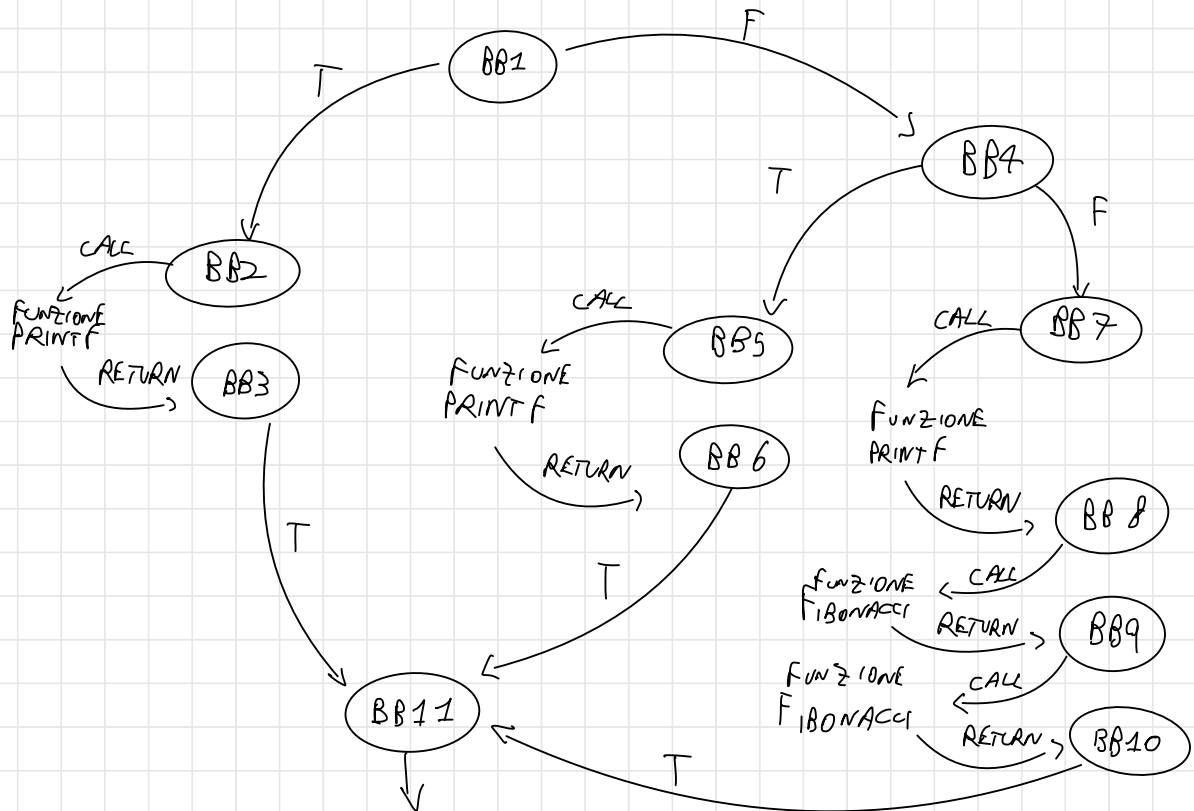
64 ; preds = %1
65 %9 = load i32, i32* %3, align 4
66 %10 = icmp eq i32 %9, 1
67     br i1 %10, label %11, label %13

68
69 ; preds = %8
70 %12 = call @i32 (@i32, ...) @printf(i8* noundef getelementptr @inbounds ([9 x i8], [9 x i8]* @.str.1, i64 0, i64 0))
71     store i32 1, i32* %2, align 4
72     br label %27

73
74 ; preds = %8
75 %14 = load i32, i32* %3, align 4
76 %15 = load i32, i32* %3, align 4
77 %16 = sub nsw i32 %15, 1
78 %17 = load i32, i32* %3, align 4
79 %18 = sub nsw i32 %17, 2
80 %19 = call @i32 (@i32, ...) @printf(i8* noundef getelementptr @inbounds ([22 x i8], [22 x i8]* @.str.2, i64 0, i64 0), i32 noundef %14, i32 noundef %16, i32
noundef %18)
81 %20 = load i32, i32* %3, align 4
82 %21 = sub nsw i32 %20, 1
83 %22 = call @Fibonacci(i32 noundef %21)
84 %23 = load i32, i32* %3, align 4
85 %24 = sub nsw i32 %23, 2
86 %25 = call @Fibonacci(i32 noundef %24)
87 %26 = add nsw i32 %22, %25
88     store i32 %26, i32* %2, align 4
89     br label %27

90
91 ; preds = %13, %11, %6
92 %28 = load i32, i32* %2, align 4
93     ret i32 %28
94 }

```



CFG PROGRAMMA FIBONACCI

