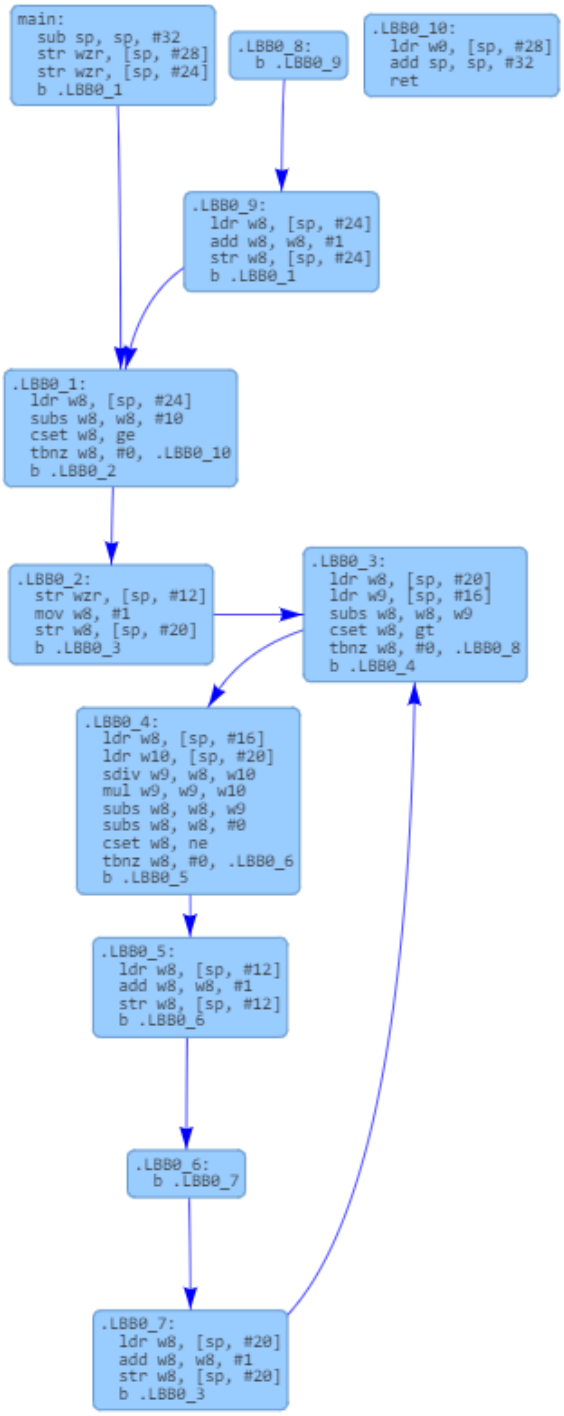
Ficha 1: Armv8-a

```
main:
  sub sp, sp, #32
  str wzr, [sp, #28]
  str wzr, [sp, #24]
  b .LBB0_1
.LBB0_1:
  ldr w8, [sp, #24]
  subs w8, w8, #10
  cset w8, ge
  tbnz w8, #0, .LBB0_10
  b .LBB0_2
.LBB0_2:
  str wzr, [sp, #12]
  mov w8, #1
  str w8, [sp, #20]
  b .LBB0_3
.LBB0_3:
  ldr w8, [sp, #20]
  ldr w9, [sp, #16]
  subs w8, w8, w9
  cset w8, gt
  tbnz w8, #0, .LBB0_8
  b .LBB0_4
.LBB0_4:
  ldr w8, [sp, #16]
  ldr w10, [sp, #20]
  sdiv w9, w8, w10
  mul w9, w9, w10
  subs w8, w8, w9
  subs w8, w8, #0
  cset w8, ne
  tbnz w8, #0, .LBB0_6
  b .LBB0_5
.LBB0_5:
  ldr w8, [sp, #12]
  add w8, w8, #1
  str w8, [sp, #12]
  b .LBB0_6
.LBB0_6:
  b .LBB0_7
.LBB0_7:
  ldr w8, [sp, #20]
  add w8, w8, #1
  str w8, [sp, #20]
  b .LBB0_3
.LBB0_8:
  b .LBB0_9
.LBB0_9:
  ldr w8, [sp, #24]
  add w8, w8, #1
  str w8, [sp, #24]
  b .LBB0_1
.LBB0_10:
  ldr w0, [sp, #28]
  add sp, sp, #32
  ret
```

```
main:
 d10083ff
sub     sp, sp, #0x20
 b9001fff
str     wzr, [sp, #28]
 b9001bff
str     wzr, [sp, #24]
 14000001
b  7c4 <main+0x10>
 b9401be8
ldr     w8, [sp, #24]
 71002908
subs    w8, w8, #0xa
 1a9fb7e8
cset    w8, ge  // ge = tcont
 37000468
tbnz    w8, #0, 85c <main+0xa8>
 14000001
b  7d8 <main+0x24>
 b9000fff
str     wzr, [sp, #12]
 52800028
mov     w8, #0x1
 b90017e8
str     w8, [sp, #20]
 14000001
b  7e8 <main+0x34>
 b94017e8
ldr     w8, [sp, #20]
 b94013e9
ldr     w9, [sp, #16]
 6b090108
subs    w8, w8, w9
```



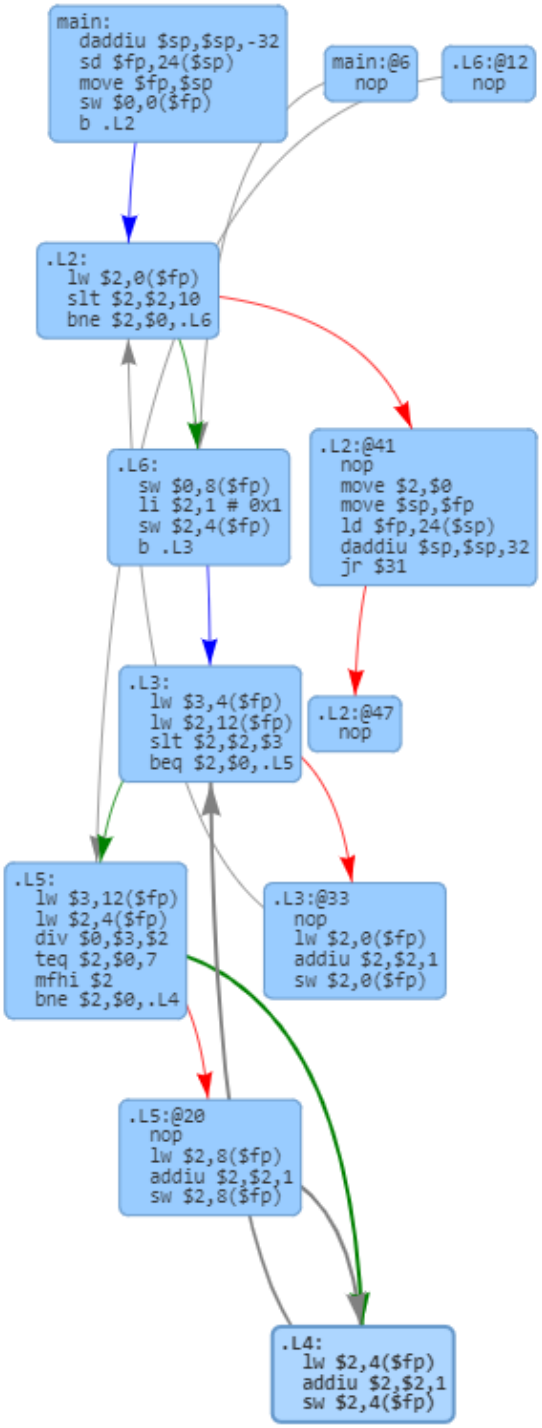
INFORMAÇÕES

- 31 registradores de 32 bits, W0..W30
- 31 registradores de 64 bits, X0..X30
- Registradores 0: WZR, XZR
- Registrador de endereço SP aponta para uma memória temporária
- Endereços de memória aparecem entre []
- Constantes são precedidas por #
- . LBB0_1 é um exemplo de rótulo para que possa ser usado no controle do fluxo do programa

Ficha 2: MIPS64

```
main:
  daddiu $sp,$sp,-32
  sd $fp,24($sp)
  move $fp,$sp
  sw $0,0($fp)
  b .L2
  nop

.L6:
  sw $0,8($fp)
  li $2,1 # 0x1
  sw $2,4($fp)
  b .L3
  nop

.L5:
  lw $3,12($fp)
  lw $2,4($fp)
  div $0,$3,$2
  teq $2,$0,7
  mfhi $2
  bne $2,$0,.L4
  nop

  lw $2,8($fp)
  addiu $2,$2,1
  sw $2,8($fp)
.L4:
  lw $2,4($fp)
  addiu $2,$2,1
  sw $2,4($fp)
.L3:
  lw $3,4($fp)
  lw $2,12($fp)
  slt $2,$2,$3
  beq $2,$0,.L5
  nop

  lw $2,0($fp)
  addiu $2,$2,1
  sw $2,0($fp)
.L2:
  lw $2,0($fp)
  slt $2,$2,10
  bne $2,$0,.L6
  nop

  move $2,$0
  move $sp,$fp
  ld $fp,24($sp)
  daddiu $sp,$sp,32
  jr $31
  nop
```

```
main:
 67bdffe0
daddiu sp,sp,-32
 ffbe0018
sd s8,24(sp)
 03a0f025
move   s8,sp
 afc00000
sw zero,0(s8)
 1000001b
b  120000a60 <main+0x80>
 00000000
nop
 afc00008
sw zero,8(s8)
 24020001
li v0,1
 afc20004
sw v0,4(s8)
 1000000e
b  120000a40 <main+0x60>
 00000000
nop
 8fc3000c
lw v1,12(s8)
 8fc20004
lw v0,4(s8)
 0062001a
div    zero,v1,v0
 004001f4
teq    v0,zero,0x7
 00001010
mfhi   v0
```



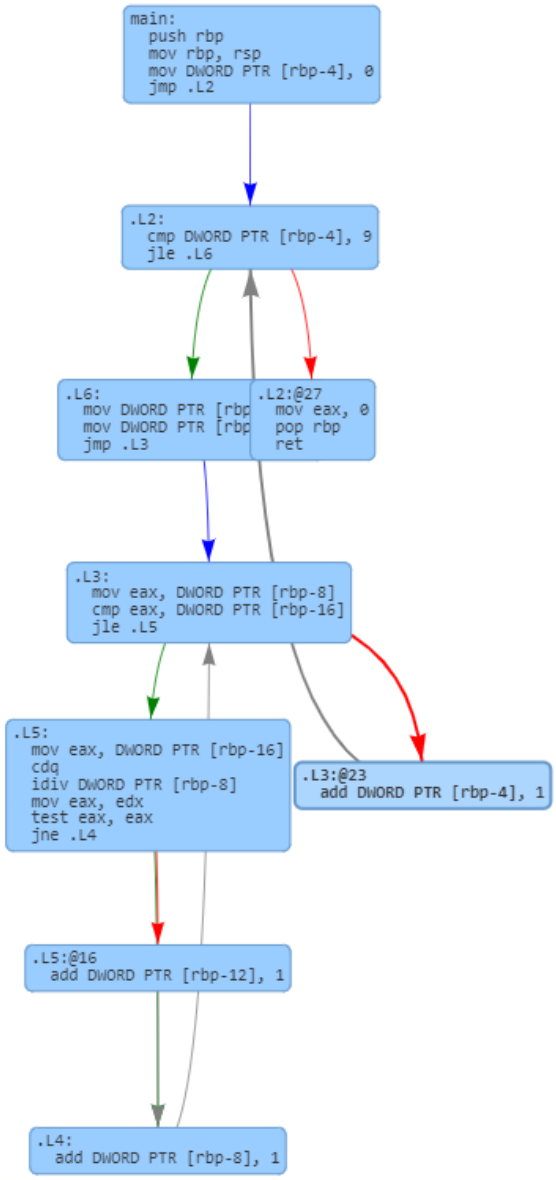
INFORMAÇÕES

- 32 registradores de 64 bits, $0...$31
- Zero == $0
- v0 e v1 são registradores para retorno de valores de funções
- Registrador de endereço s8 (equivalente a $fp) aponta para região de memória temporária de uma função
- Registrador de endereço SP aponta para uma memória temporária
- Endereços de memória aparecem entre ()
- .L6 é um exemplo de rótulo para que possa ser usado no controle do fluxo do programa

Ficha 3: x86

```
main:
  push rbp
  mov rbp, rsp
  mov DWORD PTR [rbp-4], 0
  jmp .L2
.L6:
  mov DWORD PTR [rbp-12], 0
  mov DWORD PTR [rbp-8], 1
  jmp .L3
.L5:
  mov eax, DWORD PTR [rbp-16]
  cdq
  idiv DWORD PTR [rbp-8]
  mov eax, edx
  test eax, eax
  jne .L4
  add DWORD PTR [rbp-12], 1
.L4:
  add DWORD PTR [rbp-8], 1
.L3:
  mov eax, DWORD PTR [rbp-8]
  cmp eax, DWORD PTR [rbp-16]
  jle .L5
  add DWORD PTR [rbp-4], 1
.L2:
  cmp DWORD PTR [rbp-4], 9
  jle .L6
  mov eax, 0
  pop rbp
  ret
```

```
main:
 55
push   rbp
 48 89 e5
mov    rbp,rsp
 c7 45 fc 00 00 00 00
mov    DWORD PTR [rbp-0x4],0x0
 c7 45 f8 00 00 00 00
mov    DWORD PTR [rbp-0x8],0x0
 83 7d f8 0a
cmp    DWORD PTR [rbp-0x8],0xa
 0f 8d 59 00 00 00
jge    11a5 <main+0x75>
 c7 45 ec 00 00 00 00
mov    DWORD PTR [rbp-0x14],0x0
 c7 45 f4 01 00 00 00
mov    DWORD PTR [rbp-0xc],0x1
 8b 45 f4
mov    eax,DWORD PTR [rbp-0xc]
 3b 45 f0
cmp    eax,DWORD PTR [rbp-0x10]
 0f 8f 2c 00 00 00
jg     1192 <main+0x62>
 8b 45 f0
mov    eax,DWORD PTR [rbp-0x10]
 99
cdq
 f7 7d f4
idiv   DWORD PTR [rbp-0xc]
 83 fa 00
cmp    edx,0x0
 0f 85 09 00 00 00
jne    117f <main+0x4f>
```



INFORMAÇÕES

- 16 registradores de 64 bits, rax, rbx, rcx, rdx, rbp, rsp, rsi, rdi, r8...r15
- 8 registradores de 32 bits, eax, ebx, ecx, edx, ebp, esp, esi, edi
- Endereços de memória aparecem entre []
- .L6 é um exemplo de rótulo para que possa ser usado no controle do fluxo do programa
- Constantes em hexa são precedidas por 0x

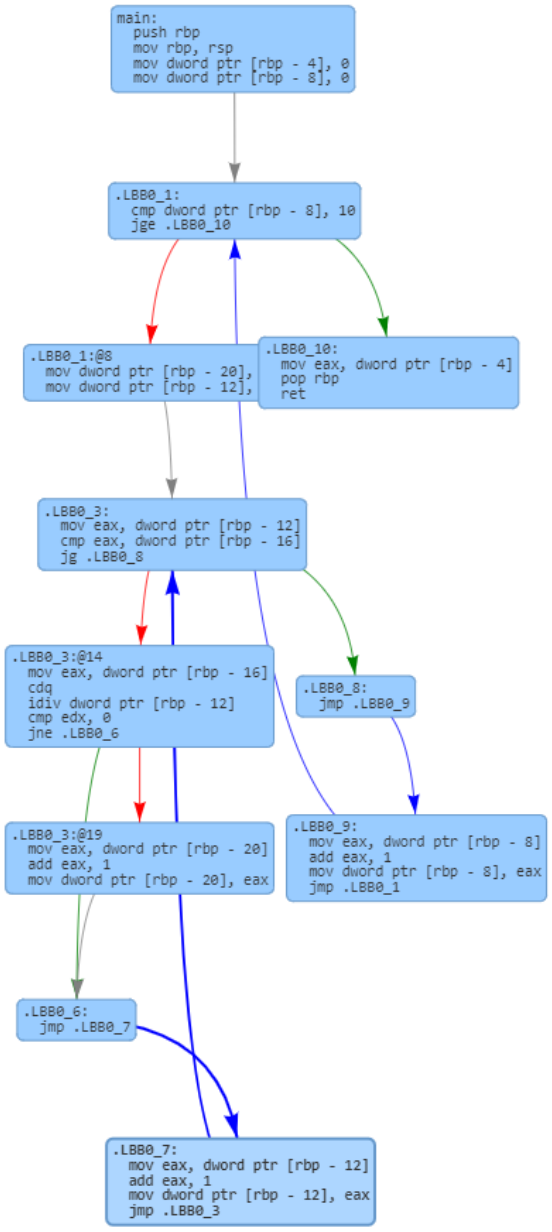
Ficha 4: x86

```
main: # @main
  push rbp
  mov rbp, rsp
  mov dword ptr [rbp - 4], 0
  mov dword ptr [rbp - 8], 0
.LBB0_1:
  cmp dword ptr [rbp - 8], 10
  jge .LBB0_10
  mov dword ptr [rbp - 20], 0
  mov dword ptr [rbp - 12], 1
.LBB0_3:
  mov eax, dword ptr [rbp - 12]
  cmp eax, dword ptr [rbp - 16]
  jg .LBB0_8
  mov eax, dword ptr [rbp - 16]
  cdq
  idiv dword ptr [rbp - 12]
  cmp edx, 0
  jne .LBB0_6
  mov eax, dword ptr [rbp - 20]
  add eax, 1
  mov dword ptr [rbp - 20], eax
.LBB0_6:
  jmp .LBB0_7
.LBB0_7:
  mov eax, dword ptr [rbp - 12]
  add eax, 1
  mov dword ptr [rbp - 12], eax
  jmp .LBB0_3
.LBB0_8:
  jmp .LBB0_9
.LBB0_9:
  mov eax, dword ptr [rbp - 8]
  add eax, 1
  mov dword ptr [rbp - 8], eax
  jmp .LBB0_1
.LBB0_10:
  mov eax, dword ptr [rbp - 4]
  pop rbp
  ret
```

```
main:
 55
push   rbp
48 89 e5
mov    rbp,rsp
c7 45 fc 00 00 00 00
mov    DWORD PTR [rbp-0x4],0x0
c7 45 f8 00 00 00 00
mov    DWORD PTR [rbp-0x8],0x0
83 7d f8 0a
cmp    DWORD PTR [rbp-0x8],0xa
0f 8d 59 00 00 00
jge    11a5 <main+0x75>
c7 45 ec 00 00 00 00
mov    DWORD PTR [rbp-0x14],0x0
c7 45 f4 01 00 00 00
mov    DWORD PTR [rbp-0xc],0x1
8b 45 f4
mov    eax,DWORD PTR [rbp-0xc]
3b 45 f0
cmp    eax,DWORD PTR [rbp-0x10]
0f 8f 2c 00 00 00
jg     1192 <main+0x62>
8b 45 f0
mov    eax,DWORD PTR [rbp-0x10]
99
cdq
f7 7d f4
idiv   DWORD PTR [rbp-0xc]
83 fa 00
cmp    edx,0x0
0f 85 09 00 00 00
jne    117f <main+0x4f>
```

INFORMAÇÕES

- 16 registradores de 64 bits, rax, rbx, rcx, rdx, rbp, rsp, rsi, rdi, r8...r15
- 8 registradores de 32 bits, eax, ebx, ecx, edx, ebp, esp, esi, edi
- Endereços de memória aparecem entre []
- .L6 é um exemplo de rótulo para que possa ser usado no controle do fluxo do programa
- Constantes em hexa são precedidas por 0x



```c
int main()
{
  int i, j, n, cont;

  for(i=0;i<10;i++){
    cont = 0;
    for (j=1;j<=n;j++){
      if ( n % j == 0 ){
         cont++;
      }
    }
  }
}
```