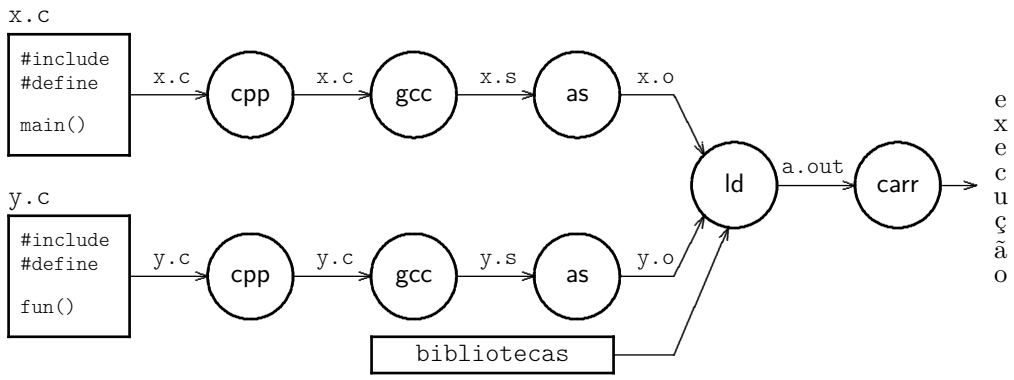

Figura 1: Etapas do processo de compilação.

Programa 1: Módulo 1, no arquivo `mod1.c`.

```
1  int k[64], l, m; // variáveis globais
2
3  int f(int, int); // definida em outro módulo
4  int g(int, int); // definida em outro módulo
5  void printf(char *, ...);
6  extern int r;    // definida em outro módulo
7
8  int main(int argc, char **argv) {
9     k[0] = 2;
10    l = f(k[1], r);
11    m = g(l, 5);
12    printf("k=%d␣l=%d␣m=%d\n", k[0], l, m);
13    return(0);
14 }
```

Programa 2: Módulo 2, no arquivo `mod2.c`.

```
1  int p, q, r;     // variáveis globais
2
3  int g(int, int); // definida em outro módulo
4  void printf(char *, ...);
5  extern int y;    // definida em outro módulo
6
7  int f(int i, int j) {
8     printf("arquivo␣2:␣i=%d␣j=%d\n", i, j);
9     return( g(i, j/y) );
10 }
```

Programa 3: Módulo 3, no arquivo `mod3.c`.

```
1  int x[128], y, z; // variáveis globais
2
3  void printf(char *, ...);
4  extern int k[];  // definida em outro módulo
5
6  int g(int i, int j) {
7     printf("arquivo␣3:␣i=%d\n", i);
8     return( i*k + k[j] );
9  }
```

Módulo 1
argc, auto, –, sp+40
argv, auto, –, sp+44
k, data, global, ?
l, data, global, ?
m, data, global, ?
main, text, global, ?
f, text, indef, ?
r, data, indef, ?
g, text, indef, ?
printf, text, indef, ?

Módulo 2
i, auto, –, sp+8
j, auto, –, sp+12
p, data, global, ?
q, data, global, ?
r, data, global, ?
y, data, indef, ?
f, text, global, ?
g, text, indef, ?
printf, text, indef, ?

Módulo 3
i, auto, –, sp+8
j, auto, –, sp+12
x, data, global, ?
y, data, global, ?
z, data, global, ?
k, data, indef, ?
g, text, global, ?
printf, text, indef, ?

Figura 2: Tabelas de símbolos dos três módulos antes da ligação.

```
# módulo 1
      .text
      .global main
      .extern r,f,g
      .extern printf
main:
000:  addi sp,sp,-64
      ...
010:  la s2, k
014:  lw a0, 4(s2)
018:  la t1, r
01c:  lw a1, 0(t1)
020:  jal f
024:  nop
028:  sw v0, 256(s2)
      ...
1f8:  jr, ra
1fc:  addi sp,sp,64

      .data
      .global k,l,m
      .org 0x800
k:    .space 0x100
l:    .space 4
m:    .space 4
```

```
# módulo 2
      .text
      .global f
      .extern y,g
      .extern printf
f:
000:  addi sp,sp,-16
      ...
020:  la s3, p
024:  lw a0, 0(s3)
028:  la t2, y
02c:  lw a1, 0(t2)
030:  jal g
034:  nop
038:  sw v0, 8(s3)
      ...
0f8:  jr, ra
0fc:  addi sp,sp,16

      .data
      .global p,q,r
      .org 0x800
p:    .space 4
q:    .space 4
r:    .space 4
```

```
# módulo 3
      .text
      .global g
      .extern k
      .extern printf
g:
000:  addi sp,sp,-32
      ...
040:  la s4, x
044:  lw a0, 512(s4)
048:  la s5, k
04c:  lw a1, 0(s5)
      ...
2f8:  jr, ra
2fc:  addi sp,sp,32

      .data
      .global x,y,z
      .org 0x800
x:    .space 0x200
y:    .space 4
z:    .space 4
```

Figura 3: Código gerado para os três módulos.

```
# TEXT segment
main: # módulo 1
1000: addi sp,sp,-64
      ...
1010: la s2, k
1014: lw a0, 4(s2)
1018: la t1, r
101c: lw a1, 0(t1)
1020: jal f
1024: nop
1028: sw v0, 256(s2)
      ...
11f8: jr, ra
11fc: addi sp,sp,64

f:    # módulo 2
1200: addi sp,sp,-16
      ...
1220: la s3, p
1224: lw a0, 0(s3)
1228: la t2, y
122c: lw a1, 0(t2)
1230: jal g
1234: nop
1238: sw v0, 8(s3)
      ...
12f8: jr, ra
12fc: addi sp,sp,16

g:    # módulo 3
1300: addi sp,sp,-32
      ...
1340: la s4, x
1344: lw a0, 512(s4)
1348: la s5, k
134c: lw a1, 0(s5)
      ...
15f8: jr, ra
15fc: addi sp,sp,32
```

```
# DATA segment
8000: .space 4 // k[64]
8100: .space 4 // l
8104: .space 4 // m

8108: .space 4 // p
810c: .space 4 // q
8110: .space 4 // r

8114: .space 4 // x[128]
8314: .space 4 // y
8318: .space 4 // z
```

Figura 4: Leiaute do arquivo objeto com os três módulos.

Módulo 1
argc, auto, –, sp+40
argv, auto, –, sp+44
k, data, –, 8000
l, data, –, 8100
m, data, –, 8104
main, text, –, 1000
f, text, –, 1200
r, data, –, 8110
g, text, –, 1300
printf, text, –, 5000

Módulo 2
i, auto, –, sp+8
j, auto, –, sp+12
p, data, –, 8108
q, data, –, 810c
r, data, –, 8110
y, data, –, 8314
f, text, –, 1200
g, text, –, 1300
printf, text, –, 5000

Módulo 3
i, auto, –, sp+8
j, auto, –, sp+12
x, data, –, 8114
y, data, –, 8314
z, data, –, 8318
k, data, –, 8000
g, text, –, 1300
printf, text, –, 5000

Figura 5: Tabelas de símbolos dos três módulos após a ligação.

endereço alto	não especificado
Bloco de Informação	cadeias dos argumentos
(com tamanho variável)	cadeias do ambiente
	informação auxiliar
	não especificado
Vetor auxiliar	elemento nulo
(elementos são doubleword)	vetor auxiliar
	palavra de zeros
Apontadores para ambiente	uma endereço para cada elemento
	palavra de zeros
Apontadores para argumentos	uma endereço para cada elemento
sp →	número de argumentos ‡
	indefinido

‡ alinhado como *double-word*

Figura 6: Registro de ativação de `main()`.

Exemplo A pilha inicial de um programa que é invocado com `'cp src dst'` é mostrada na Figura 7. A pilha é alocada em `0x7fc0.0000`, o programa recebe duas variáveis de ambiente `HOME=/home/dir` e `PATH=/home/dir/bin:/usr/bin`, e seu vetor auxiliar é inicializado com o descritor de arquivo número 13 (`a_type = AT_EXECFD = 2`). Note que `argv[argc]==0`.

	b0 b1 b2 b3	*quatro bytes numa palavra*
	n \0 *pad pad*	*endereço alto*
	r / b i	
	: / u s	
0x7fbf.fff0	/ b i n	
	/ d i r	
	h o m e	
	T H = /	
0x7fbf.ffe0	r \0 P A	PATH=/home/dir/bin:/usr/bin\0
	e / d i	
	/ h o m	
	0 M E =	
0x7fbf.ffd0	s t \0 H	HOME=/home/dir\0
	r c \0 d	
	c p \0 s	cp\0 src\0 dst\0
	0	
0x7fbf.ffc0	0	vet_aux[1] = {0,0} (elmto nulo)
	13	
	2	vet_aux[0] = {2,13} (file descr)
	0	separador
0x7fbf.ffb0	0x7fbf.ffe2	"PATH=..."
	0x7fbf.ffd3	vetor de ambiente: "HOME=..."
	0	separador, argv[argc]==0
	0x7fbf.ffcf	"dst"
0x7fbf.ffa0	0x7fbf.ffcb	"src"
	0x7fbf.ffc8	vetor de argumentos: "cp"
sp→	3	número de argumentos

Figura 7: Pilha inicial do processo `"cp src dst"`.