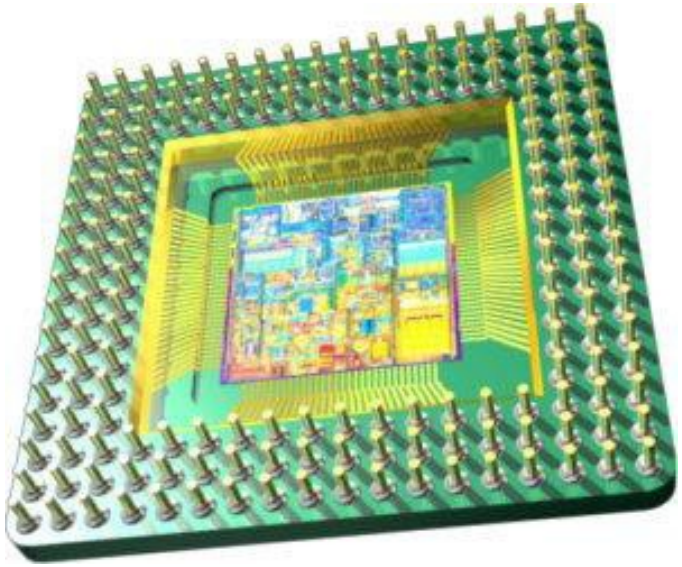
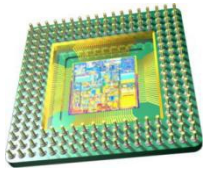


Sistemas de Microprocessadores (DEEC)

Arquitectura de Computadores (DEI)

Aulas de Laboratório

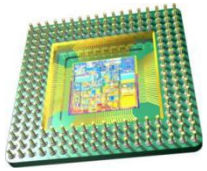
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA



LAB1 – Linguagem C e Compilação de Programas

- Sumário:

1. Etapas de compilação de um programa escrito em linguagem C.
2. `gcc`: comandos de compilação e *flags* principais.
3. Compilação e teste de programas muito simples.
4. Organização de programas em vários ficheiros de código fonte e compilação incremental.
5. Utilização de *makefiles*.



Etapas de Compilação

código fonte

```
void main(){
  int a=1,b=2,c;

  c=a+b;
}
```

main.c

Pré-Processamento

código fonte pré-processado

Compilação

código Assembly

main.s

Assembling

main.o

código objecto

Linking

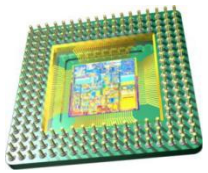
main.exe

código executável

```
gcc main.c -o main
```

.c , .s : ficheiro ASCII (editável)

.o , .exe : ficheiro binário (não editável)



Etapas de compilação: Pré-Processam.

```
void main(){
  int a=1,b=2,c;

  c=a+b;
}
```

main.c

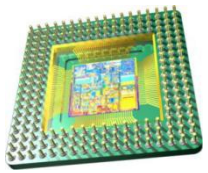
As linguagens de alto nível são independentes da arquitetura e do SO.

Pré-processamento

`gcc -E main.c`

```
# 1 "main.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "main.c"
void main()
{
int a=1, b=2, c;
c=a+b;
}
```

Pré-processamento de directivas de compilação (ex. **`#include`**, **`#define`**, etc.).



Etapas de compilação: Compilação

```
void main(){
   int a=1,b=2,c;

   c=a+b;
}
```

main.c

As linguagens de alto nível são independentes da arquitetura e do SO.

Compilação

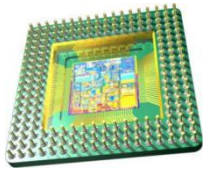
```
gcc -S main.c
```

```
.file       1 "main.c"
.section    .mdebug.abi32
.previous
.abicalls
.text
.align      2
.globl      main
.ent        main
.type       main, @function
main:
.frame      $fp,32,$31
.mask       0x40000000,-8
.fmask      0x00000000,0
.set        noreorder
.set        nomacro
addiu       $sp,$sp,-32
sw          $fp,24($sp)
move        $fp,$sp
li          $2,1                    #0x1
sw          $2,16($fp)
li          $2,2                    #0x2
sw          $2,12($fp)
lw          $3,16($fp)
lw          $2,12($fp)
nop
addu        $2,$3,$2
sw          $2,8($fp)
move        $sp,$fp
lw          $fp,24($sp)
addiu       $sp,$sp,32
j $31
nop

.set        macro
.set        reorder
.end        main
.ident      "GCC: (GNU) 4.1.2 20061115"
```

main.s

A linguagem Assembly depende da arquitetura do computador (ex. MIPS).



Etapas de compilação: Assembling

```
.file "main.c"
.text
.globl main
.type main, @function
main:
.LFB0:
.cfi_startproc
pushq %rbp
.cfi_def_cfa_offset 16
movq %rsp, %rbp
.cfi_offset 6, -16
.cfi_def_cfa_register 6
movl -8(%rbp), %eax
movl -4(%rbp), %edx
leal (%rdx,%rax), %eax
movl %eax, -12(%rbp)
leave
ret
.cfi_endproc
.LFE0:
.size     main, .-main
.ident    "GCC: (Ubuntu 4.4.3-4ubuntu5)"
.section .note.GNU-stack,"",@progbits
```

Assembling

```
gcc main.s –o main
```

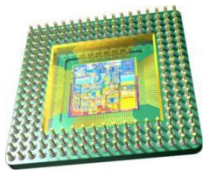
```
00000000  fc 31 c0 8e c0 8e d8 8e  d0 bc 00 7c 89 e6 bf 00
00000010  06 b9 00 01 f3 a5 89 fd  b1 08 f3 ab fe 45 f2 e9
00000020  00 8a f6 46 bb 20 75 08  84 d2 78 07 80 4e bb 40
00000030  8a 56 ba 88 56 00 e8 fc  00 52 bb c2 07 31 d2 88
00000040  6f fc 0f a3 56 bb 73 19  8a 07 bf 87 07 b1 03 f2
00000050  ae 74 0e b1 0b f2 ae 83  c7 09 8a 0d 01 cf e8 c5
00000060  00 42 80 c3 10 73 d8 58  2c 7f 3a 06 75 04 72 05
00000070  48 74 0d 30 c0 04 b0 88  46 b8 bf b2 07 e8 a6 00
00000080  be 7b 07 e8 b2 00 8a 56  b9 4e e8 8e 00 eb 05 b0
00000090  07 e8 b0 00 30 e4 cd 1a  89 d7 03 7e bc b4 01 cd
000000a0  16 75 0d 30 e4 cd 1a 39  fa 72 f2 8a 46 b9 eb 16
000000b0  30 e4 cd 16 88 e0 3c 1c  74 f1 2c 3b 3c 04 76 06
000000c0  2c c7 3c 04 77 c9 98 0f  a3 46 0c 73 c2 88 46 b9
000000d0  be 00 08 8a 14 89 f3 3c  04 9c 74 0a c0 e0 04 05
000000e0  be 07 93 c6 07 80 53 f6  46 bb 40 75 08 bb 00 06
000000f0  b4 03 e8 59 00 5e 9d 75  06 8a 56 b8 80 ea 30 bb
00000100  00 7c b4 02 e8 47 00 72  86 81 bf fe 01 55 aa 0f
00000110  85 7c ff be 85 07 e8 19  00 ff e3 b0 46 e8 24 00
00000120  b0 31 00 d0 eb 17 0f ab  56 0c be 78 07 e8 eb ff
00000130  89 fe e8 03 00 be 85 07  ac a8 80 75 05 e8 04 00
00000140  eb f6 24 7f 53 bb 07 00  b4 0e cd 10 5b c3 8a 74
00000150  01 8b 4c 02 b0 01 56 89  e7 f6 46 bb 80 74 13 66
00000160  6a 00 66 ff 74 08 06 53  6a 01 6a 10 89 e6 48 80
00000170  cc 40 cd 13 89 fc 5e c3  20 20 a0 0a 44 65 66 61
00000180  75 6c 74 3a a0 0d 8a 00  05 0f 01 06 07 0b 0c 0e
00000190  83 a5 a6 a9 0d 0c 0b 0a  09 08 0a 0e 11 10 01 3f
000001a0  bf 44 4f d3 4c 69 6e 75  f8 46 72 65 65 42 53 c4
000001b0  66 bb 44 72 69 76 65 20  00 00 80 8f b6 00 00 00
000001c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

`main.s`

A linguagem Assembly depende da arquitetura do computador (ex. MIPS).

`main`

Programa executável: cada instrução em código máquina (ex. código de 32 bits no MIPS).



Compilação Incremental

```
#include "soma.h"
int main(){
int a,b,c;
c=soma(a,b);
return c;
}
```

main.c

```
int soma(int a, int b)
{ return a+b; }
```

soma.c (+ soma.h)

Compilação + Assembling

`gcc -c main.c`

main.o

61

62 00000005 66B8020000000
63 0000000B 66BB01000000
64 00000011 6601D8

65
66

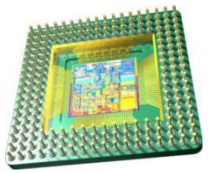
Compilação + Assembling

`gcc -c soma.c`

soma.o

Código objecto/máquina:

- Instruções codificadas em "opcodes";
- Alguns endereços por resolver; só depois de tudo ligado é que sabemos onde fica.



Etapas de compilação: Linking

main.o

soma.o

Linking

```
gcc main.o soma.o –o main
```

código objecto

61

62 00000005 66B8020000000
63 0000000B 66BB01000000
64 00000011 6601D8

65
66

```
00000000  fc 31 c0 8e c0 8e d8 8e  d0 bc 00 7c 89 e6 bf 00
00000010  06 b9 00 01 f3 a5 89 fd  b1 08 f3 ab fe 45 f2 e9
00000020  00 8a f6 46 bb 20 75 08  84 d2 78 07 80 4e bb 40
00000030  8a 56 ba 88 56 00 e8 fc  00 52 bb c2 07 31 d2 88
00000040  6f fc 0f a3 56 bb 73 19  8a 07 bf 87 07 b1 03 f2
00000050  ae 74 0e b1 0b f2 ae 83  c7 09 8a 0d 01 cf e8 c5
00000060  00 42 80 c3 10 73 d8 58  2c 7f 3a 06 75 04 72 05
00000070  48 74 0d 30 c0 04 b0 88  46 b8 bf b2 07 e8 a6 00
00000080  be 7b 07 e8 b2 00 8a 56  b9 4e e8 8e 00 eb 05 b0
00000090  07 e8 b0 00 30 e4 cd 1a  89 d7 03 7e bc b4 01 cd
000000a0  16 75 0d 30 e4 cd 1a 39  fa 72 f2 8a 46 b9 eb 16
000000b0  30 e4 cd 16 88 e0 3c 1c  74 f1 2c 3b 3c 04 76 06
000000c0  2c c7 3c 04 77 c9 98 0f  a3 46 0c 73 c2 88 46 b9
000000d0  be 00 08 8a 14 89 f3 3c  04 9c 74 0a c0 e0 04 05
000000e0  be 07 93 c6 07 80 53 f6  46 bb 40 75 08 bb 00 06
000000f0  b4 03 e8 59 00 5e 9d 75  06 8a 56 b8 80 ea 30 bb
00000100  00 7c b4 02 e8 47 00 72  86 81 bf fe 01 55 aa 0f
00000110  85 7c ff be 85 07 e8 19  00 ff e3 b0 46 e8 24 00
00000120  b0 31 00 d0 eb 17 0f ab  56 0c be 78 07 e8 eb ff
00000130  89 fe e8 03 00 be 85 07  ac a8 80 75 05 e8 04 00
00000140  eb f6 24 7f 53 bb 07 00  b4 0e cd 10 5b c3 8a 74
00000150  01 8b 4c 02 b0 01 56 89  e7 f6 46 bb 80 74 13 66
00000160  6a 00 66 ff 74 08 06 53  6a 01 6a 10 89 e6 48 80
00000170  cc 40 cd 13 89 fc 5e c3  20 20 a0 0a 44 65 66 61
00000180  75 6c 74 3a a0 0d 8a 00  05 0f 01 06 07 0b 0c 0e
00000190  83 a5 a6 a9 0d 0c 0b 0a  09 08 0a 0e 11 10 01 3f
000001a0  bf 44 4f d3 4c 69 6e 75  f8 46 72 65 65 42 53 c4
000001b0  66 bb 44 72 69 76 65 20  00 00 80 8f b6 00 00 00
000001c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

main

Programa executável: cada instrução em código máquina (ex. código de 32 bits no MIPS).