PSI3441 – Arquitetura de Sistemas Embarcados

- Assembly vs C

Escola Politécnica da Universidade de São Paulo

Prof. Gustavo Rehder – grehder@lme.usp.br

Prof. Sergio Takeo – kofuji@usp.br

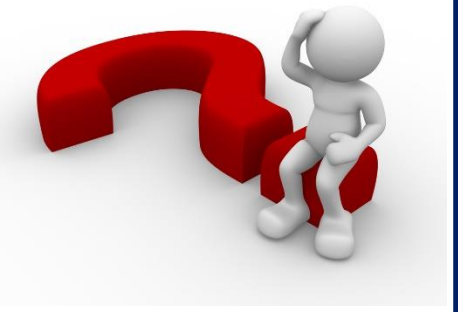
Prof. Antonio Seabra – acseabra@lsi.usp.br

2016

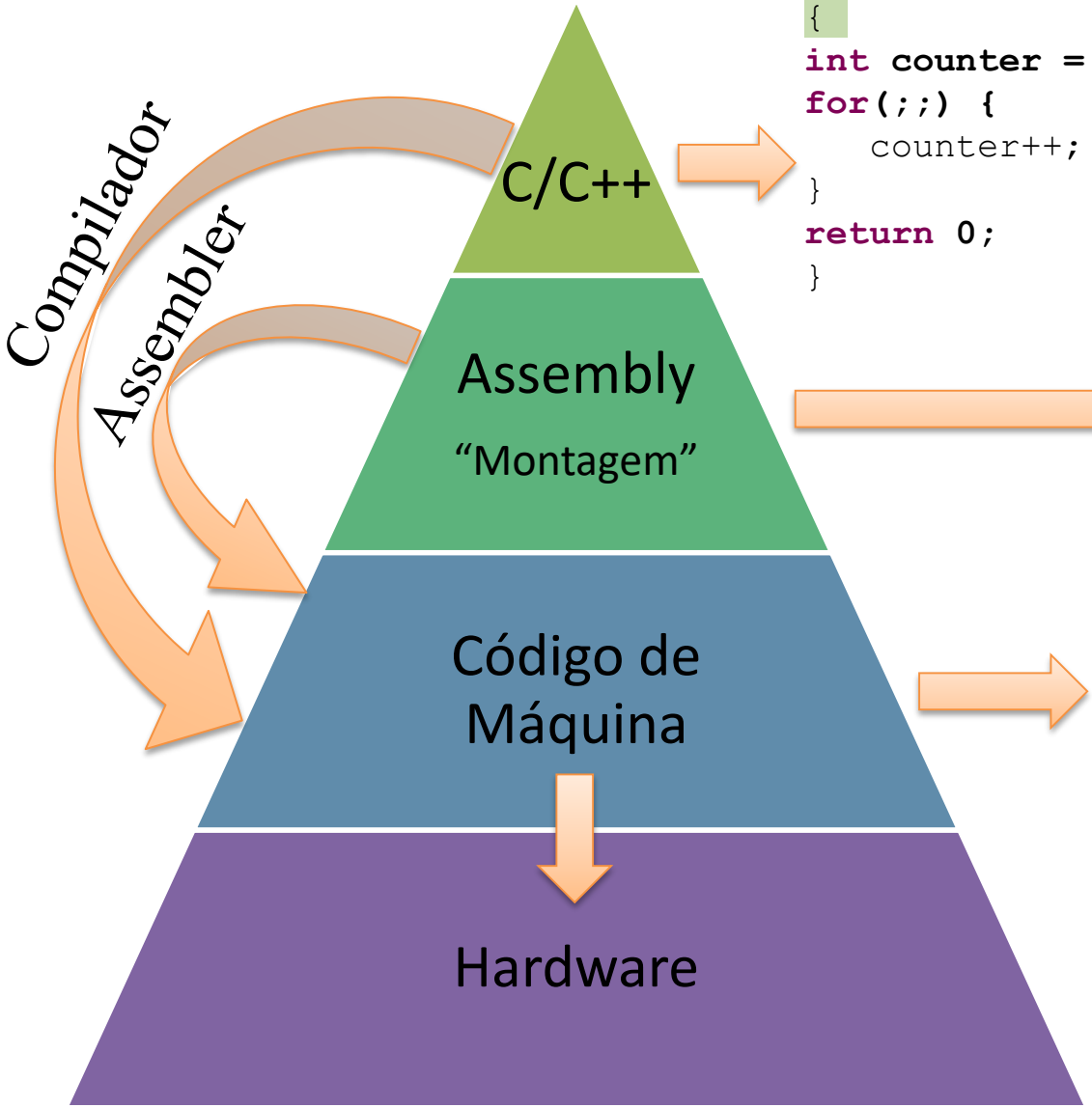
Pergunta

1. Em qual linguagem programar microcontroladores?
   - C (mais usado) e C++ (menos usado)
   - Assembly
2. Qual é a diferença entre C e Assembly?
   - Nível
   - Dificuldade/Esforço/Tempo
   - Eficiência
   - Necessidade de conhecimento do Hardware

Mais nos próximos slides...



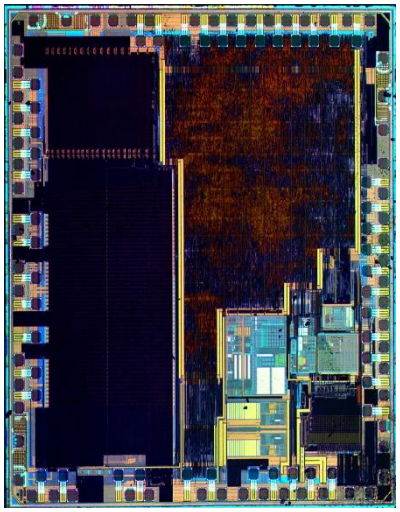
Níveis de Linguagem




```
int main(void)
{
int counter = 0;
for(;;) {
    counter++;
}
return 0;
}
```

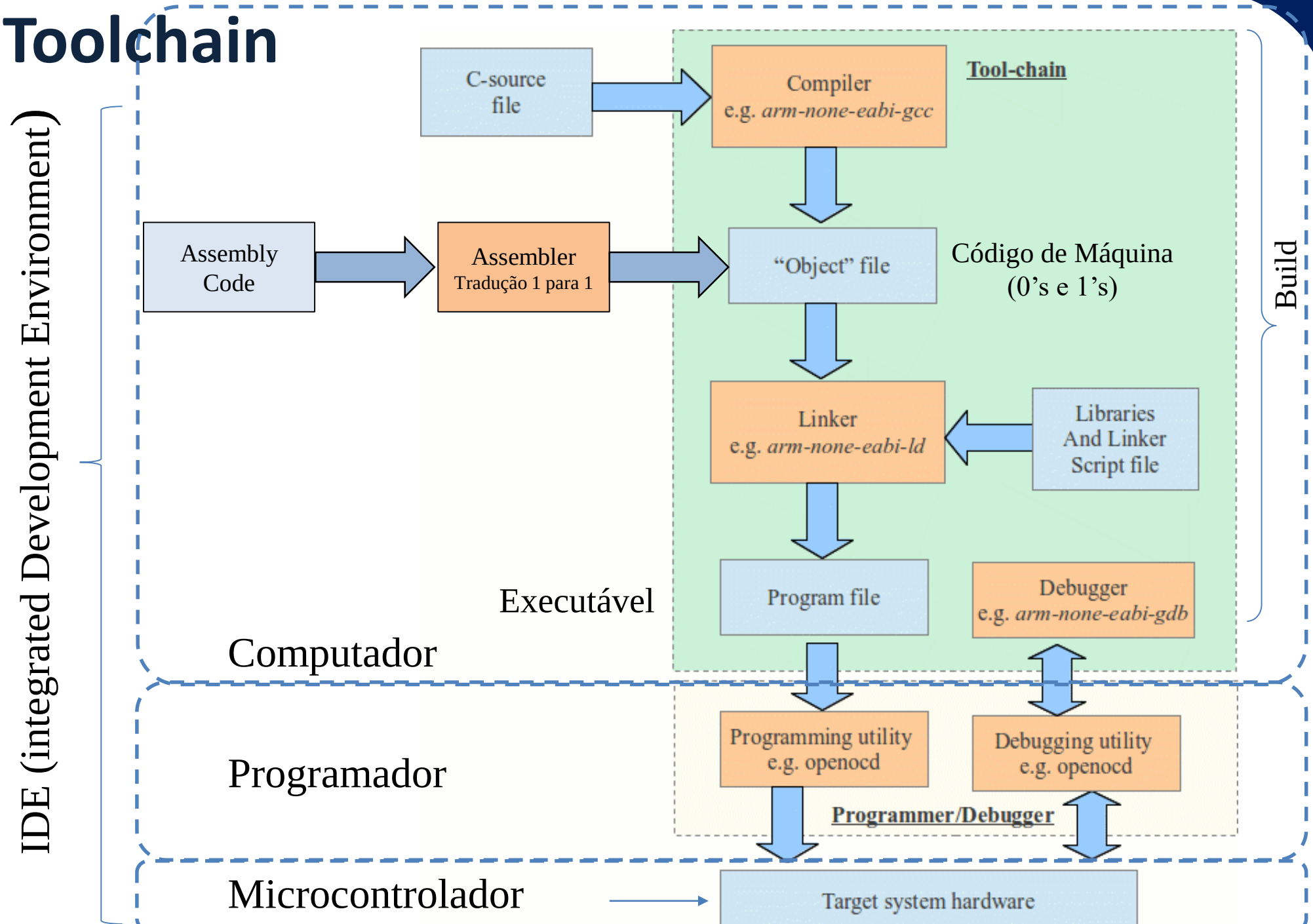
```
push {r7,lr}
sub sp,sp,#8
add r7,sp,#0
movs r3,#0
str r3,[r7,#4]
ldr r3,[r7,#4]
adds r3,#1
str r3,[r7,#4]
```

```
0x80402DE9
0x08D04DE2
0x00708DE2
0x0030B0E3
0x043087E5
0x043097E5
0x013093E2
0x043087E5
```

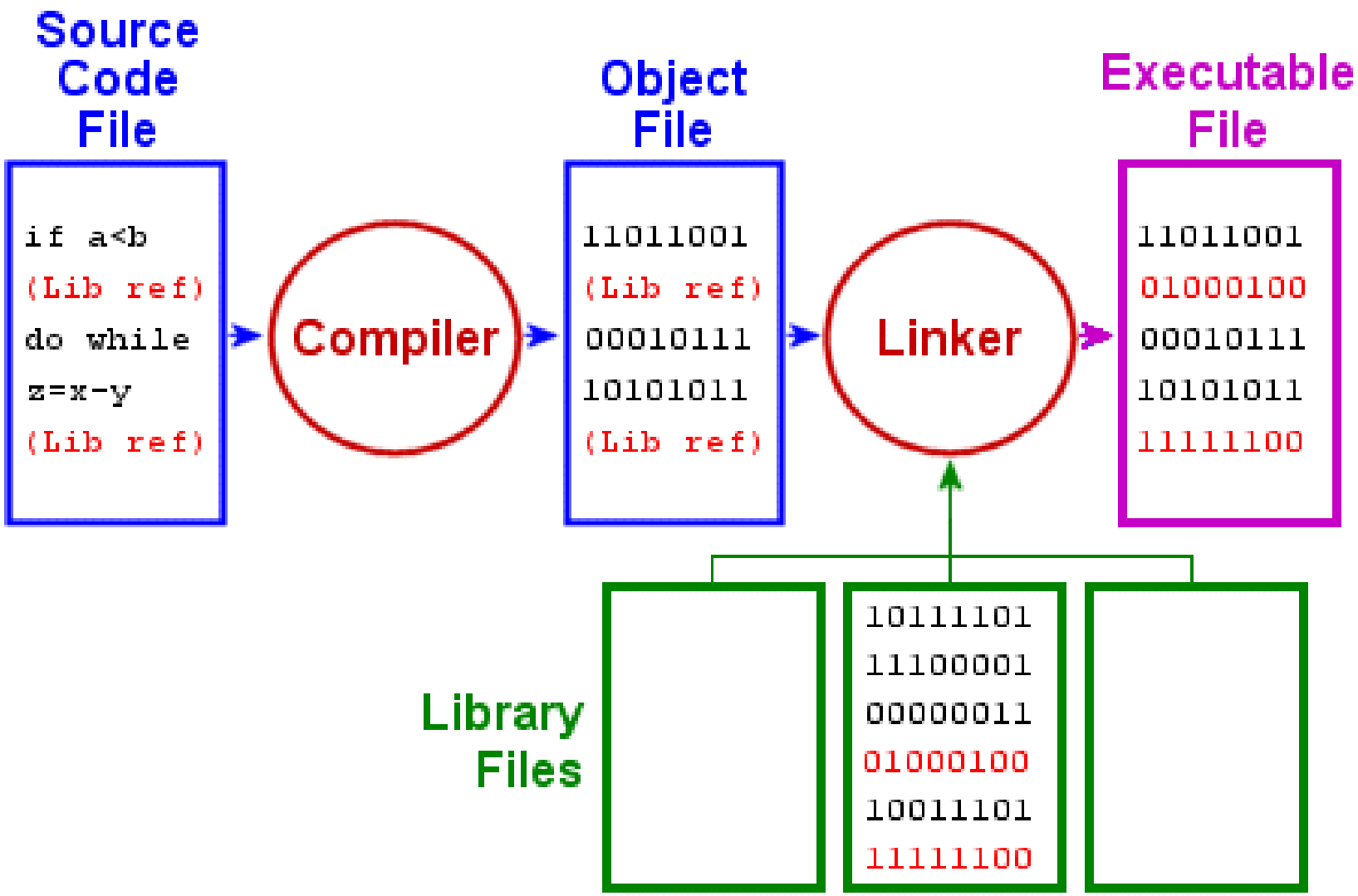


ARM Cortex M3 (STM32F100C4T6B)

Toolchain


IDE (integrated Development Environment)
C-source file
Compiler
e.g. arm-none-eabi-gcc
Tool-chain
Assembly Code
Assembler
Tradução 1 para 1
"Object" file
Código de Máquina
(0's e 1's)
Build
Linker
e.g. arm-none-eabi-ld
Libraries And Linker Script file
Executável
Program file
Debugger
e.g. arm-none-eabi-gdb
Computador
Programming utility
e.g. openocd
Debugging utility
e.g. openocd
Programmer/Debugger
Programador
Microcontrolador
Target system hardware

Exemplo de Criação de Arquivo Executável

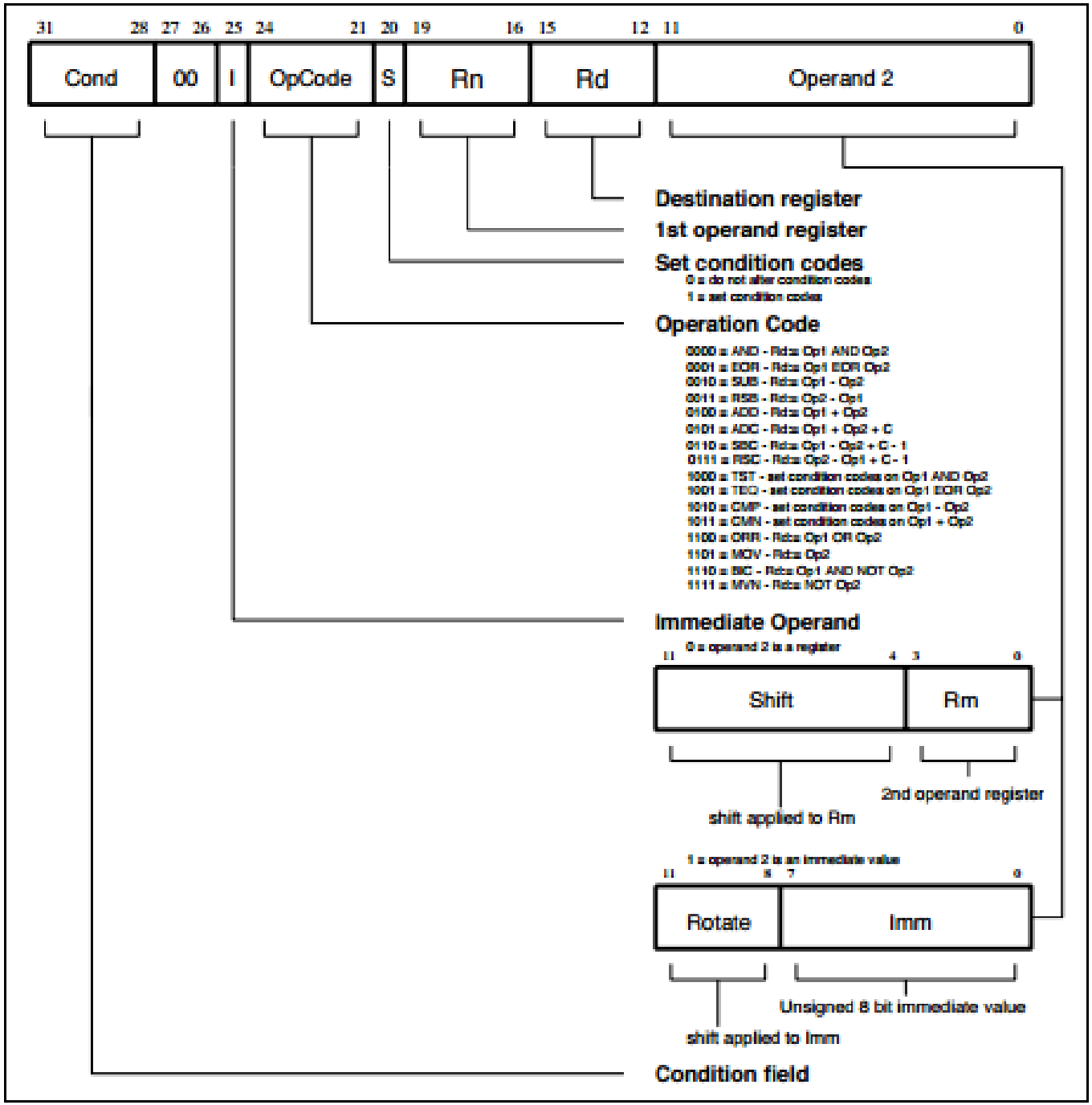


Comparação de alguns Toolchains

Companhia	atollic	Green Hills SOFTWARE	IAR SYSTEMS	ARM KEIL Microcontroller Tools	NXP
Software	Atollic TrueSTUDIO Pro	Green Hills Software Multi IDE	IAR Embedded Workbench for ARM (EWARM)	Keil PRO Edition Microcontroller Development Kit (MDK)	Kinetis Design Studio
Free version / Limitations	TrueSTUDIO Lite: Unlimited	Evaluation: 30 days	Evaluation: 30-day KickStart Edition: 32KB	MDK Lite: 32KB	Unlimited
IDE Framework	Improved/Simplified Eclipse	Proprietary	Proprietary	Proprietary	Eclipse
Debugger	GDB + proprietary extensions	Multi	IAR C-SPY®	uVison®	GDB
Compiler	Atollic GNU gcc v4.7.3	Multi	IAR icc	armcc	GNU gcc 4.8 GNU Tools for ARM Embedded (launchpad) 4.8 - Q3 2014
Standard Libraries	newlib v1.19 newlib-nano v1.0 libstdc++ v6.0.17	Multi	IAR DLIB	ARM MicroLib ARM Standard	newlib 2.1.0 newlib-nano 2.1
Run Control Interfaces	P&E, SEGGER, CMSIS-DAP (coming soon), gdbserver compatible probes	GHS Probe, GHS SuperTrace™ Probe, OpenOCD, CMSIS-DAP (coming soon)	i-jet™, P&E, SEGGER, OpenOCD, CMSIS-DAP	ULINK™, ULINKpro™, CMSIS-DAP, P&E, SEGGER	P&E, SEGGER, OpenOCD/CMSIS-DAP
Other RTOS Support Includes	FreeRTOS™, Micrium® µC/OS®	u-velOSity™	FreeRTOS, Micrium®µC/OS	FreeRTOS, Keil RTX	FreeRTOS, Micrium® µC/OS

Assembly
↓
Código de Máquina

- Específico para cada microcontrolador ou família
- Necessita conhecimento da microarquitetura do controlador
- "Tradução" direta para código de máquina
- Programação em baixo nível - mais trabalhoso
- **Mais eficiente!!!**

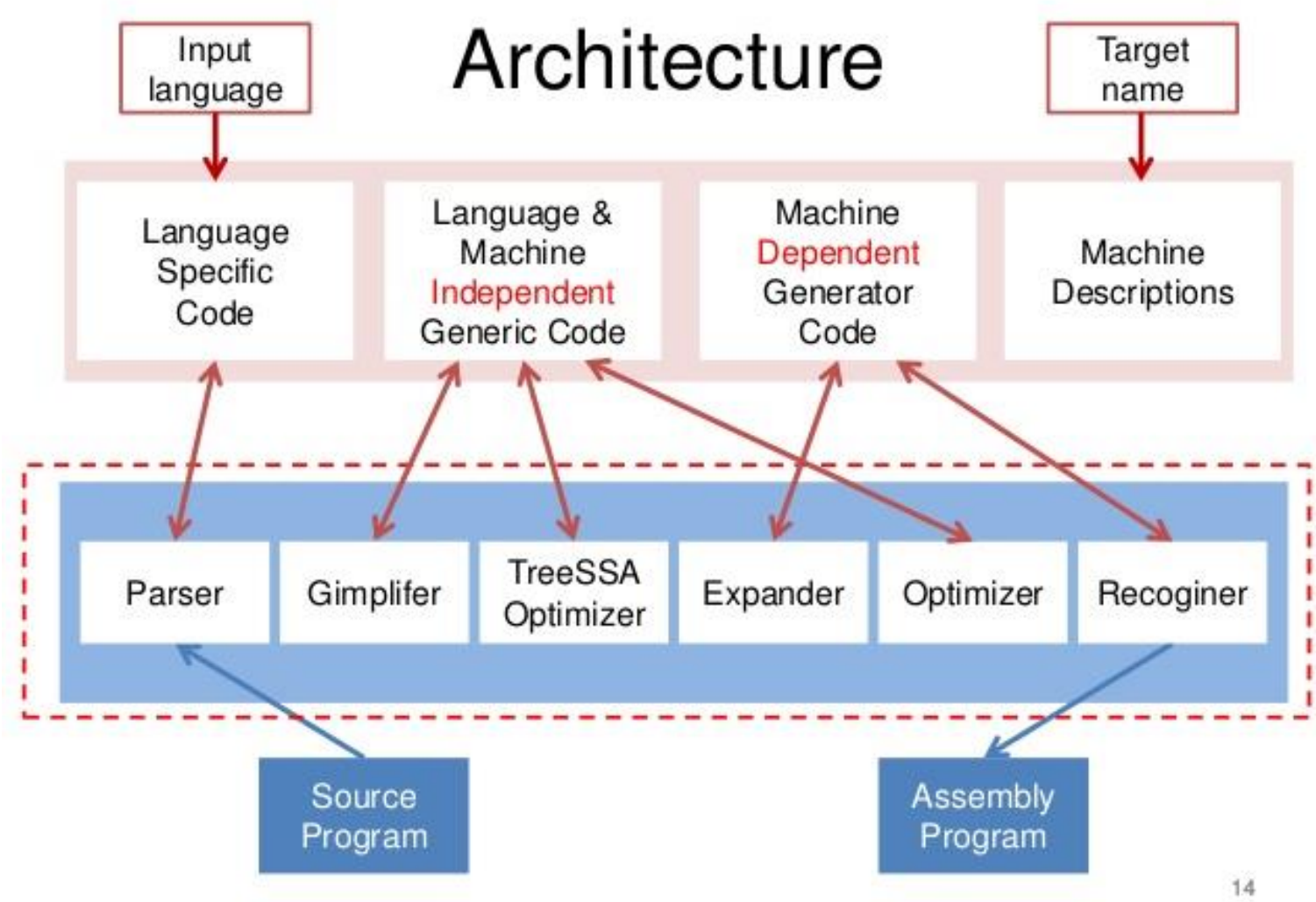


Online Assembler: http://armconverter.com/

Fonte: ARM7TDMI-S Data Sheet

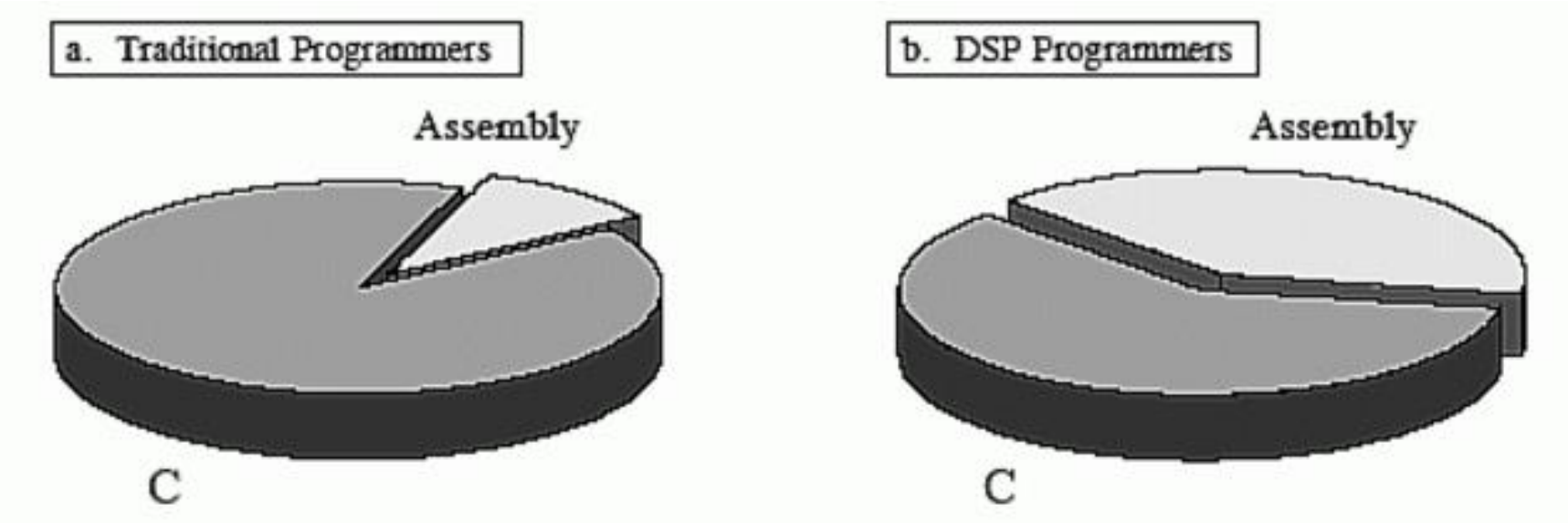
C → Código de Máquina

Compilador GCC - GNU C Compiler



- Não Necessita conhecimento da microarquitetura do controlador
- Compilação é complexa
- Código compilado pode não ser o mais eficiente
- **Programação em alto nível – mais rápido**

Quando usar Assembly?



DSP – Digital Signal Processor – Processadores otimizados para processamento de sinais de áudio e vídeo em tempo real.

Produto Escalar em C

Produto Escalar

$$\boldsymbol{x}[n] \cdot \boldsymbol{y}[n] \rightarrow x[1] \cdot y[1] + x[2] \cdot y[2] + \cdots + x[n] \cdot y[n]$$

```c
#define LEN 20
  float dm x[LEN];
  float pm y[LEN];
  float result;

main()

{
  int n;
  float s;
  for (n=0;n<LEN;n++)
    s += x[n]*y[n];
  result = s
}
```

Disassembly do Código em C compilado pelo GCC

```
push {r7,lr}
sub sp,sp,#8
add r7,sp,#0
 movs r3,#0
str r3,[r7,#4]
b main+0x6 (0xfb2)
ldr r3,[r7,#4]
adds r3,#1
str r3,[r7,#4]
ldr r3,[r7,#4]
cmp r3,#19
ble main+0x6 (0xfac)
ldr r3,[pc,#44]
ldr r2,[r7,#4]
lsls r2,r2,#2
ldr r2,[r2,r3]
ldr r3,[pc,#40]
ldr r1,[r7,#4]
  lsls r1,r1,#2
  ldr r3,[r1,r3]
  mov r0,r2
  mov r1,r3
  bl __aeabi_fmul (0xc6c)
  mov r3,r0
  ldr r0,[r7,#0]
  mov r1,r3
  bl __aeabi_fadd (0x8c8)
  mov r3,r0
  str r3,[r7,#0]
  ldr r3,[pc,#16]
  ldr r2,[r7,#0]
  str r2,[r3,#0]
```

Produto Escalar em Assembly

para Analog Device SHARC DSP

±1 ciclo de clock por linha

```
i12 = _y;                                  /* i12 points to beginning of y[ ] */
i4 = _x;                                   /* i4 points to beginning of x[ ] */

lcntr = 20, do (pc,4) until lce;           /* loop for the 20 array entries */
  f2 = dm(i4,m6);                          /* load the x[ ] value into register f2 */
  f4 = pm(i12,m14);                        /* load the y[ ] value into register f4 */
  f8 = f2*f4;                              /* multiply the two values, store in f8 */
  f12 = f8 + f12;                          /* add the product to the accumulator in f12 */

dm(_result) = f12;                         /* write the accumulator to memory */
```

4 ciclos

Código Otimizado

```
i12 = _y;                                  /* i12 points to beginning of y[ ] */
i4 = _x;                                   /* i4 points to beginning of x[ ] */

f2 = dm(i4,m6), f4 = pm(i12,m14)           /* prime the registers */
f8 = f2*f4, f2 = dm(i4,m6), f4 = pm(i12,m14);

lcntr = 18, do (pc,1) until lce;           /* highly efficient main loop */
f12 = f8 + f12, f8 = f2*f4, f2 = dm(i4,m6), f4 = pm(i12,m14);

f12 = f8 + f12, f8 = f2*f4;                /* complete the last loop */
f12 = f8 + f12;

dm(_result) = f12;                         /* store the result in memory */
```

1 ciclo