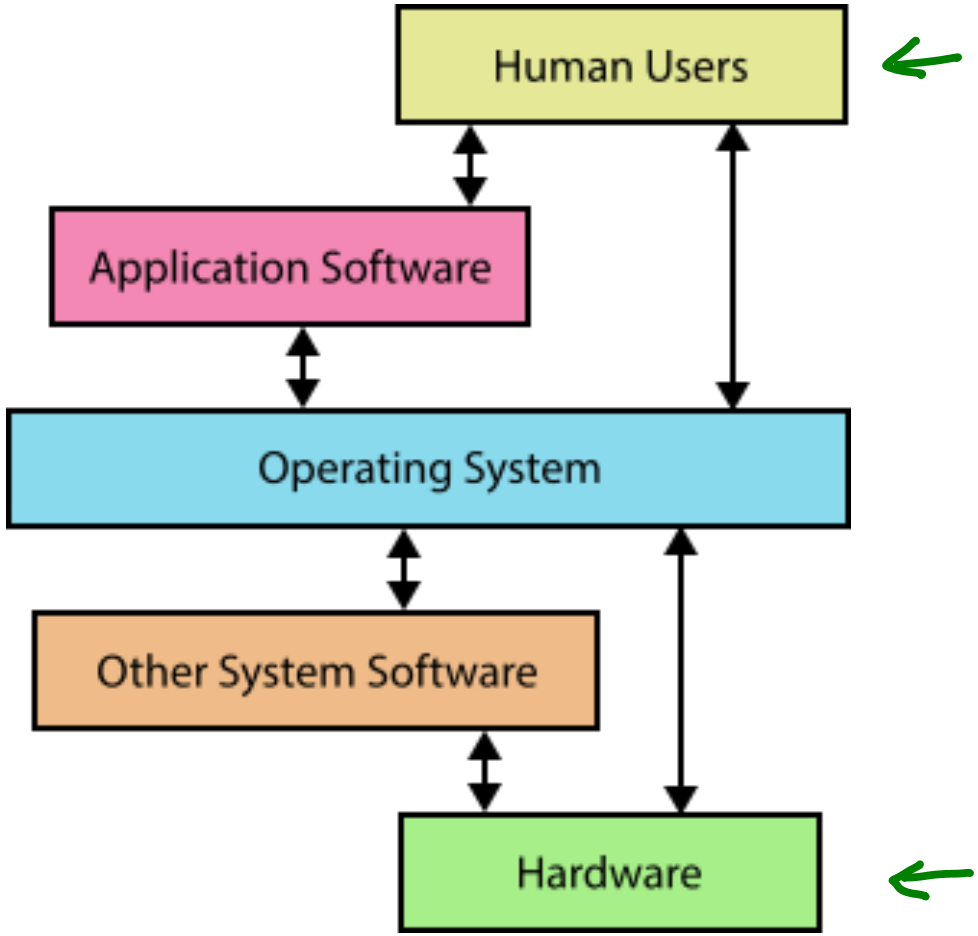
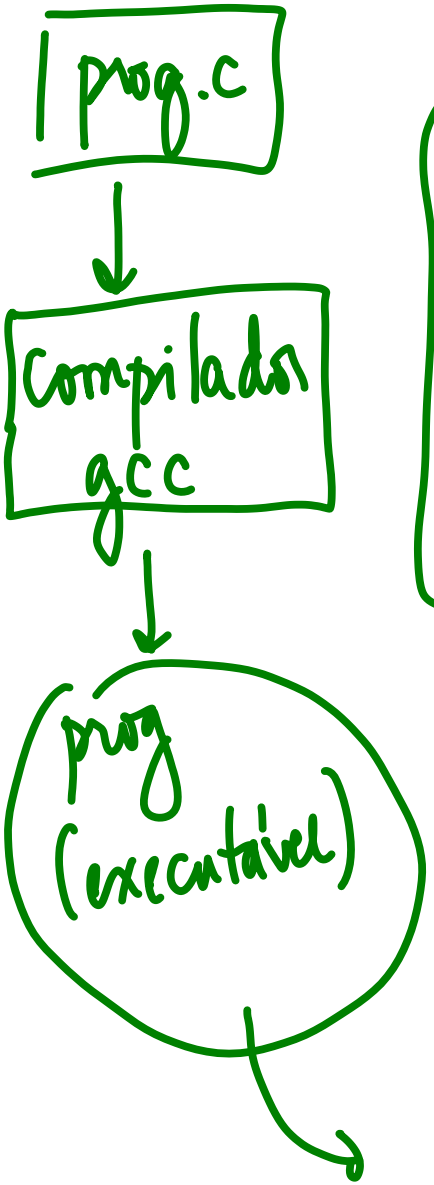
MAC0329 - 16.06.2020

um pouco sobre organização de computadores


Human Users
Application Software
Operating System
Other System Software
Hardware

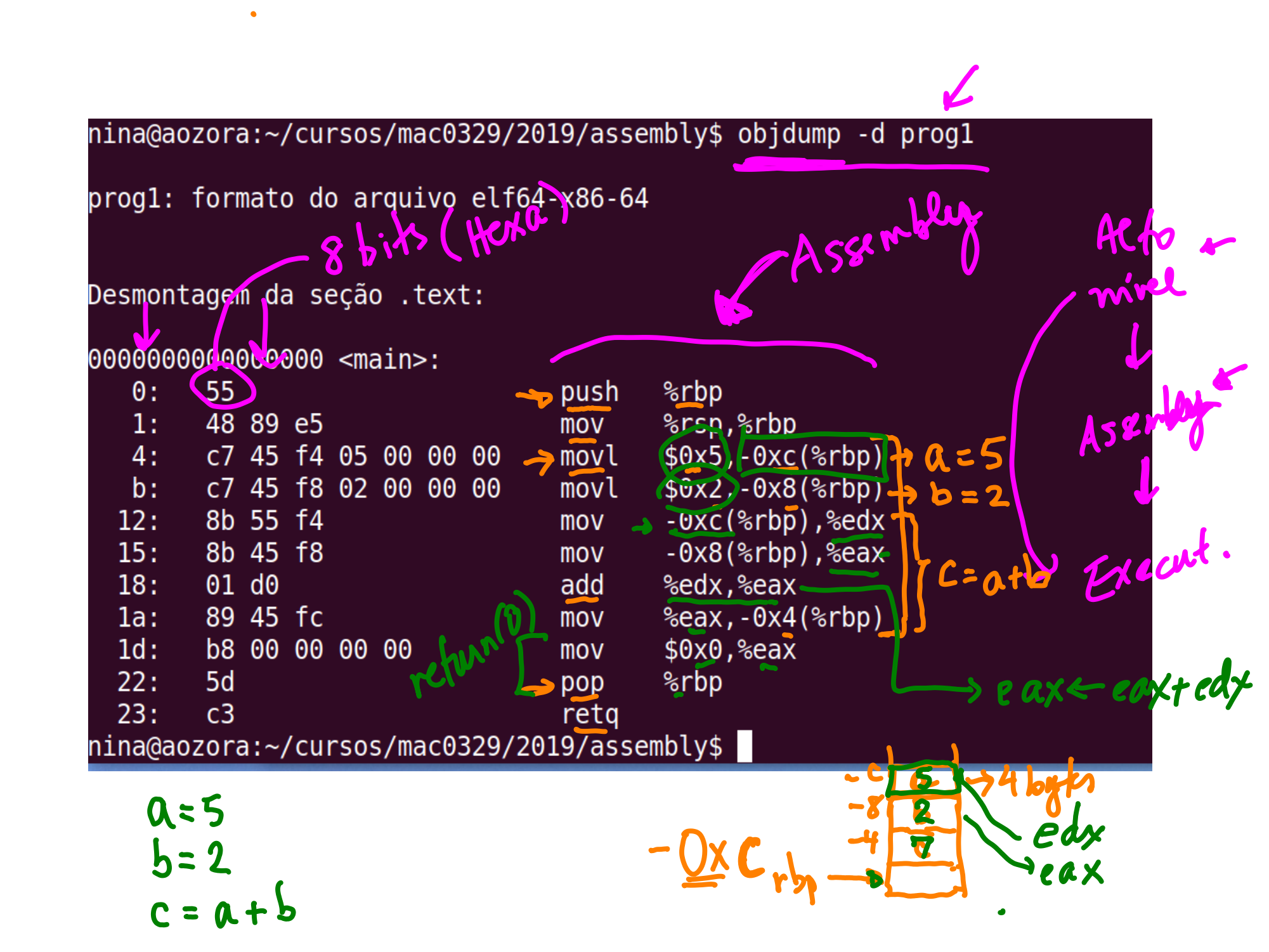



```c
#include <stdio.h>

int main() {

  int a, b, c;

  a = 5 ;

  b = 2 ;

  c = a + b ;

  return 0 ;

}

/*
   gcc -g -c prog1.c
   objdump -d -M intel -S prog1.o
*/
```

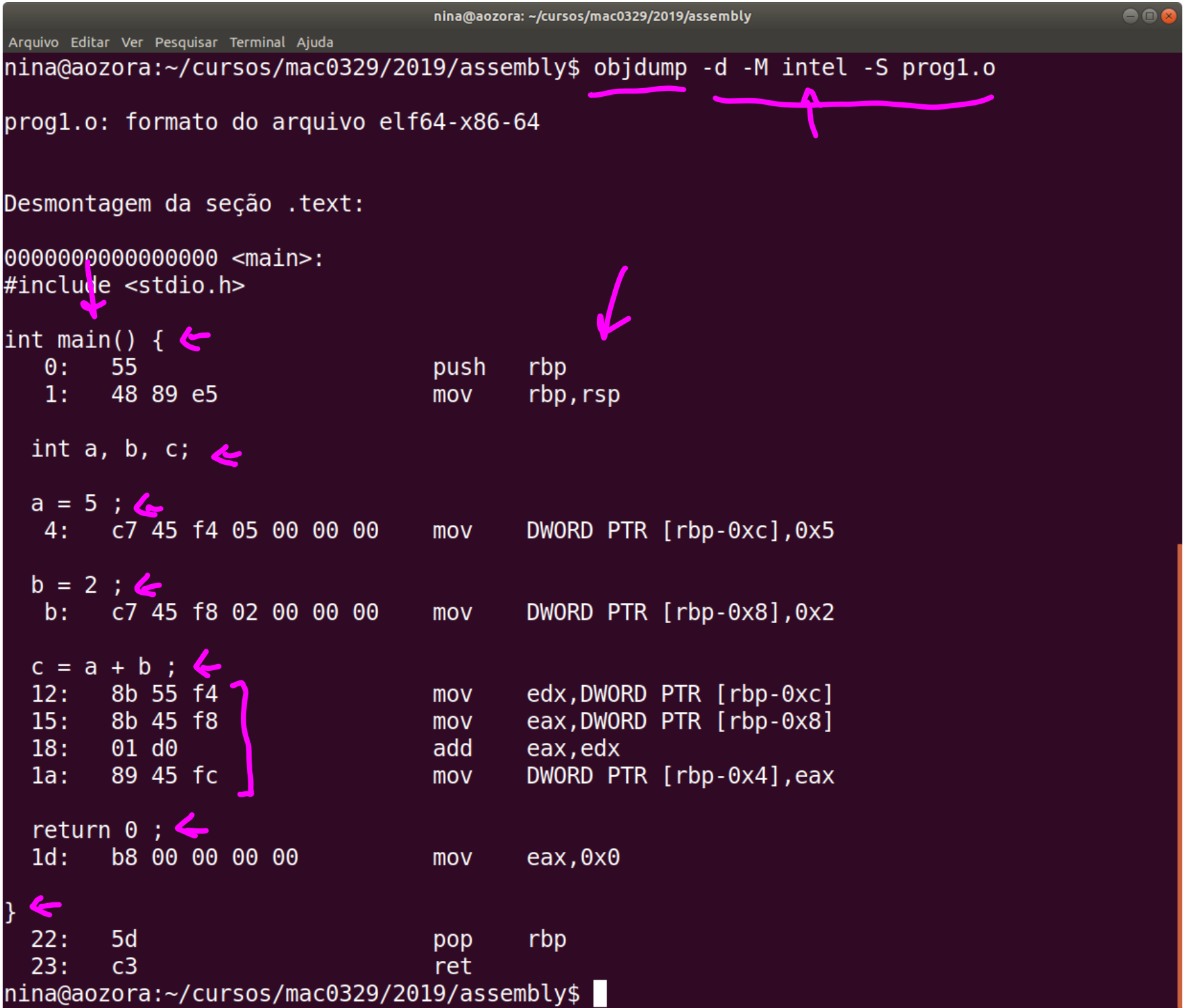
código

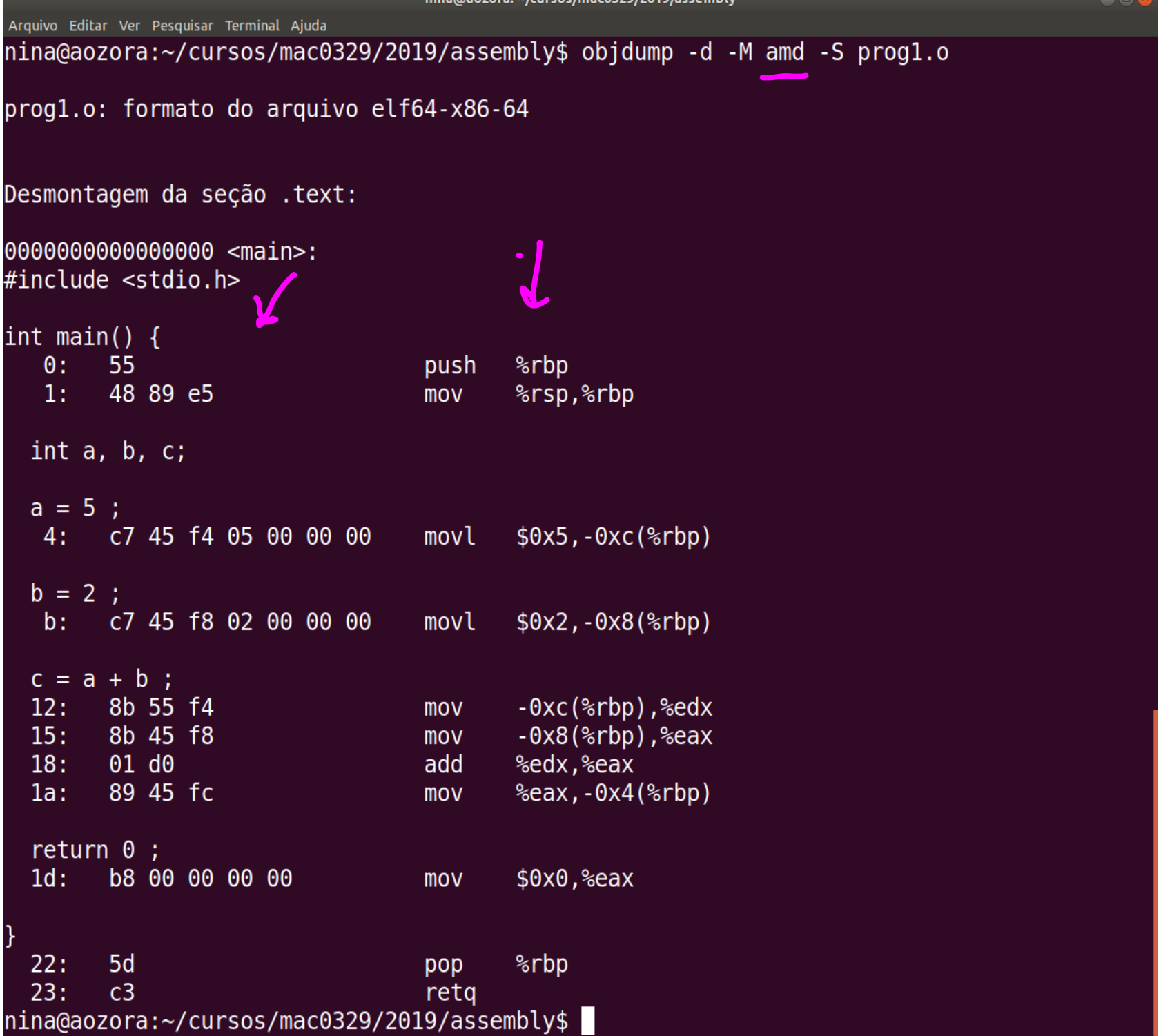
programa em linguagem de alto nível

```
nina@aozora:~/cursos/mac0329/2019/assembly$ objdump -d prog1

prog1: formato do arquivo elf64-x86-64


Desmontagem da seção .text:

0000000000000000 <main>:
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
   4:   c7 45 f4 05 00 00 00    movl   $0x5,-0xc(%rbp)
   b:   c7 45 f8 02 00 00 00    movl   $0x2,-0x8(%rbp)
  12:   8b 55 f4                mov    -0xc(%rbp),%edx
  15:   8b 45 f8                mov    -0x8(%rbp),%eax
  18:   01 d0                   add    %edx,%eax
  1a:   89 45 fc                mov    %eax,-0x4(%rbp)
  1d:   b8 00 00 00 00          mov    $0x0,%eax
  22:   5d                      pop    %rbp
  23:   c3                      retq
nina@aozora:~/cursos/mac0329/2019/assembly$
```



a=5
b=2
c=a+b


nina@aozora: ~/cursos/mac0329/2019/assembly
Arquivo Editar Ver Pesquisar Terminal Ajuda
nina@aozora:~/cursos/mac0329/2019/assembly$ objdump -d -M intel -S prog1.o
prog1.o: formato do arquivo elf64-x86-64
Desmontagem da seção .text:
0000000000000000 <main>:
#include <stdio.h>
int main() {
0: 55 push rbp
1: 48 89 e5 mov rbp,rsp
int a, b, c;
a = 5 ;
4: c7 45 f4 05 00 00 00 mov DWORD PTR [rbp-0xc],0x5
b = 2 ;
b: c7 45 f8 02 00 00 00 mov DWORD PTR [rbp-0x8],0x2
c = a + b ;
12: 8b 55 f4 mov edx,DWORD PTR [rbp-0xc]
15: 8b 45 f8 mov eax,DWORD PTR [rbp-0x8]
18: 01 d0 add eax,edx
1a: 89 45 fc mov DWORD PTR [rbp-0x4],eax
return 0 ;
1d: b8 00 00 00 00 mov eax,0x0
}
22: 5d pop rbp
23: c3 ret
nina@aozora:~/cursos/mac0329/2019/assembly$

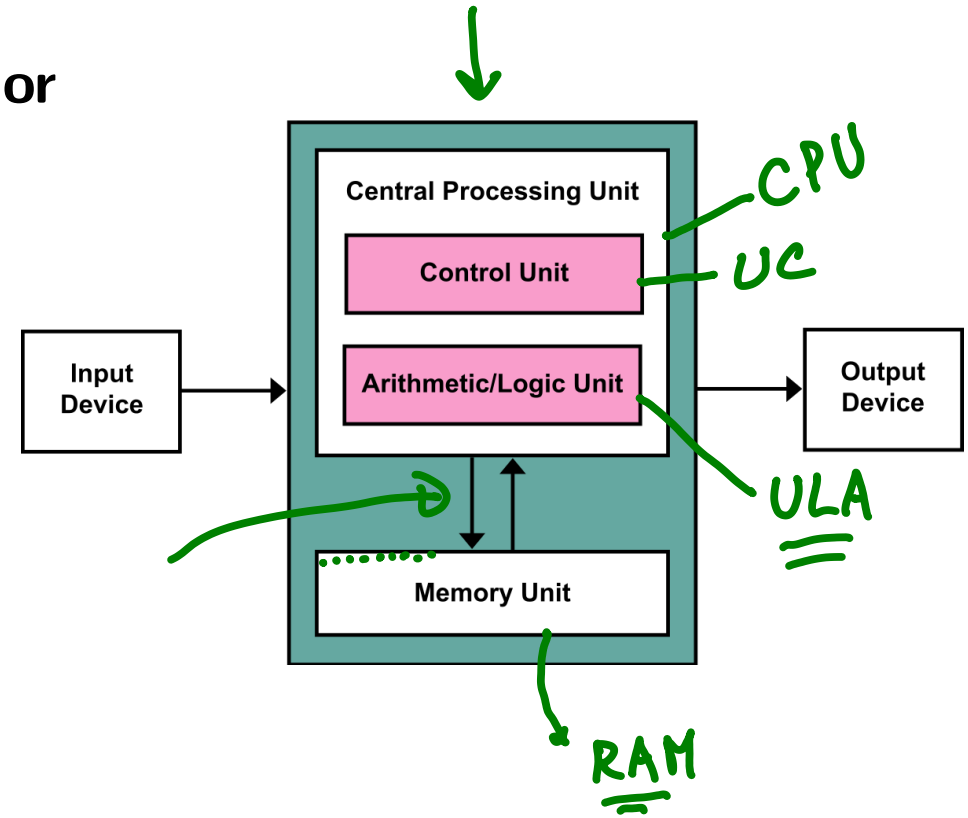

Arquivo Editar Ver Pesquisar Terminal Ajuda
nina@aozora:~/cursos/mac0329/2019/assembly$ objdump -d -M amd -S prog1.o
prog1.o: formato do arquivo elf64-x86-64
Desmontagem da seção .text:
0000000000000000 <main>:
#include <stdio.h>
int main() {
0: 55 push %rbp
1: 48 89 e5 mov %rsp,%rbp
int a, b, c;
a = 5 ;
4: c7 45 f4 05 00 00 00 movl $0x5,-0xc(%rbp)
b = 2 ;
b: c7 45 f8 02 00 00 00 movl $0x2,-0x8(%rbp)
c = a + b ;
12: 8b 55 f4 mov -0xc(%rbp),%edx
15: 8b 45 f8 mov -0x8(%rbp),%eax
18: 01 d0 add %edx,%eax
1a: 89 45 fc mov %eax,-0x4(%rbp)
return 0 ;
1d: b8 00 00 00 00 mov $0x0,%eax
}
22: 5d pop %rbp
23: c3 retq
nina@aozora:~/cursos/mac0329/2019/assembly$

Código baixo nível

Um pouco sobre organização de computadores

Partes de um computador

1. ULA
2. Unidade de controle
3. Memória (RAM)



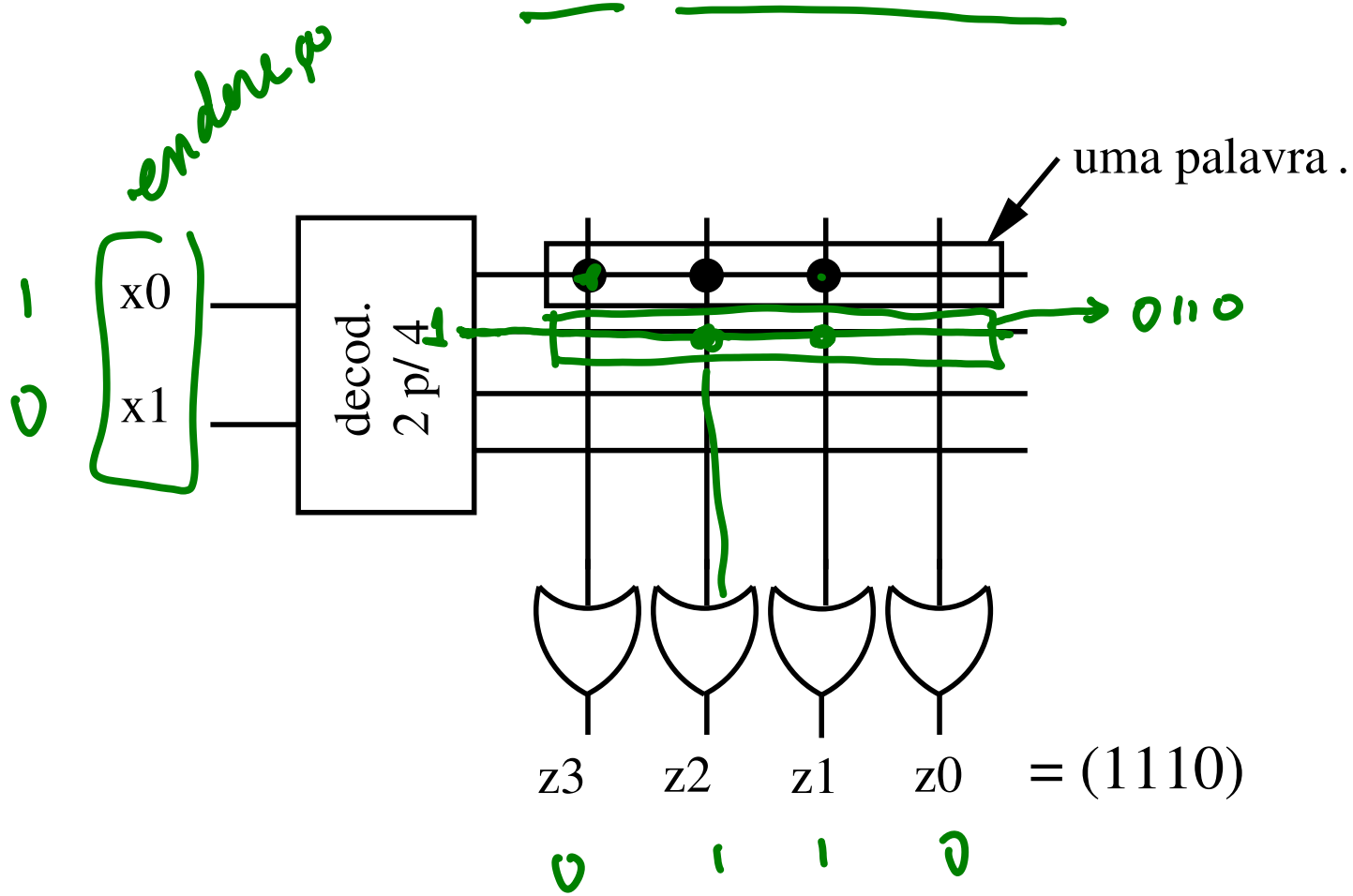
ULA

Unidade lógico-aritmética

- operações aritméticas $+ - * /$
- operações lógicas
  - $>$ $\neq$
  - $<$
  - $=$
  - $\vdots$

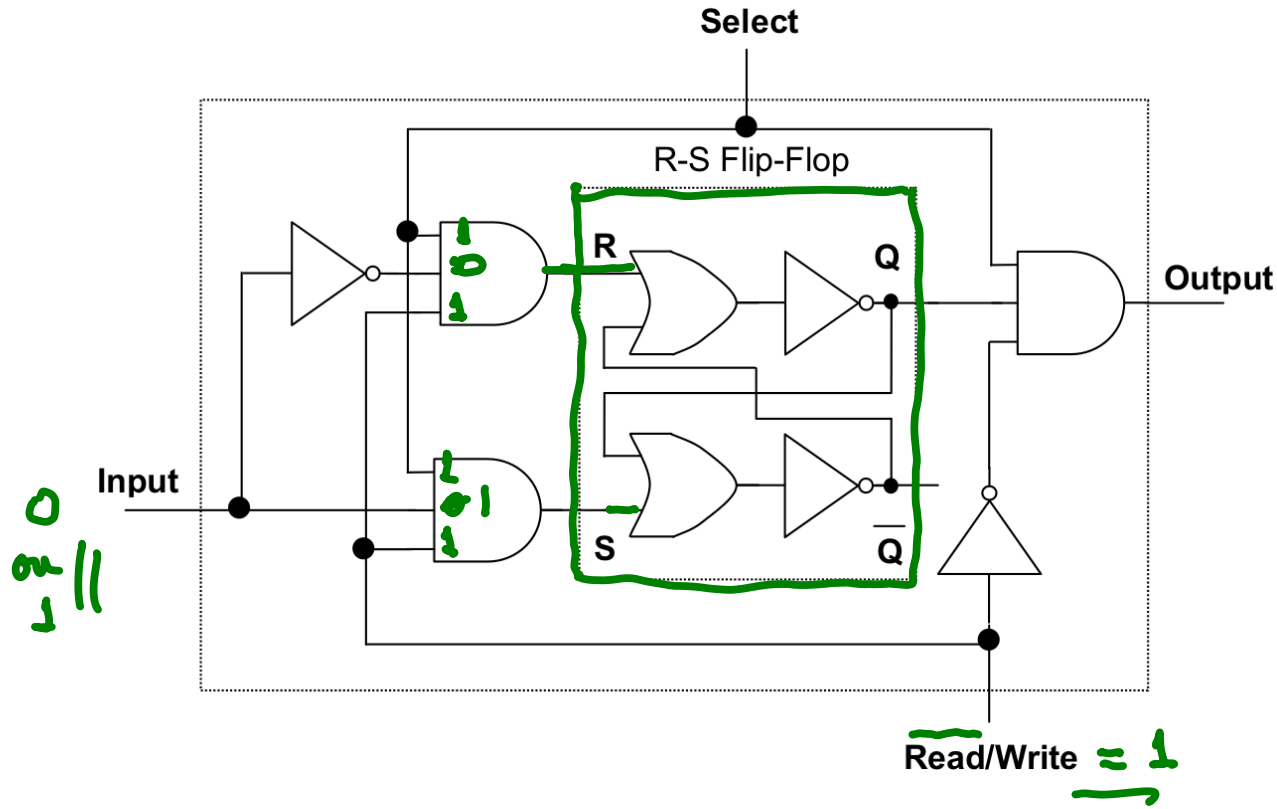
RAM - Random Access Memory

Vimos anteriormente a ROM (Read-only memory)



1 bit na RAM (é possível ler/escrever):

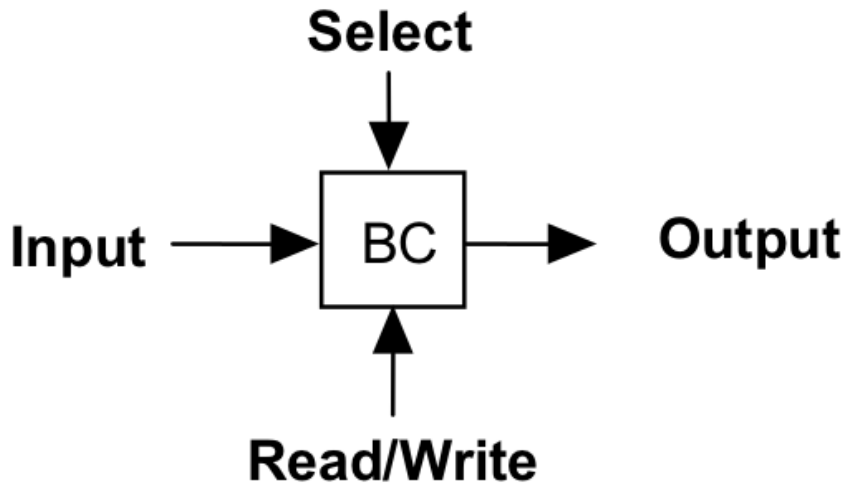
Random Access Memory



- $\overline{Read}/Write = 0 \Longrightarrow$ leitura $\quad \overline{Read}/Write = 1 \Longrightarrow$ escrita
- $select = 1$ célula fica habilitada para a operação de $Read/Write$
- $input = 1$: se $\overline{Read}/Write = 1$ e $select = 1$ a entrada set do flip-flop SR (estado do SR vai para 1)
- $input = 0$: se $\overline{Read}/Write = 1$ e $select = 1$ a entrada reset do flip-flop SR (estado do SR vai para 0)

Fonte: http://watson.latech.edu/book/circuits/circuitsMicrocomputer3.html

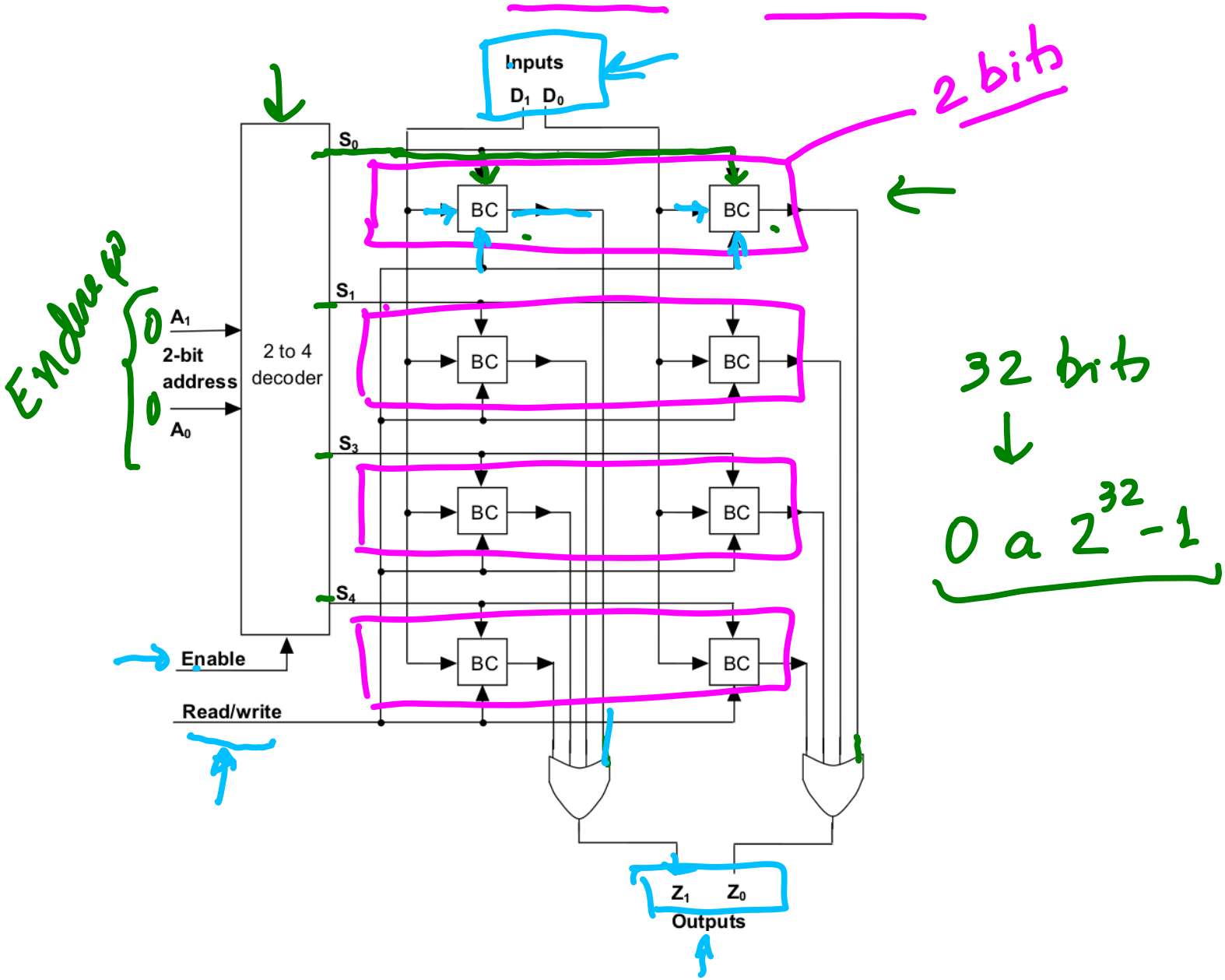
Visão caixa-preta de uma célula de bit:



- $\overline{Read}/Write = 0 \Longrightarrow$ leitura $\quad \overline{Read}/Write = 1 \Longrightarrow$ escrita
- $select = 1$ célula fica habilitada para a operação de $\overline{Read}/Write$
- $input = 1$: se $\overline{Read}/Write = 1$ e $select = 1$ a entrada set do flip-flop SR (estado do SR vai para 1)
- $input = 0$: se $\overline{Read}/Write = 1$ e $select = 1$ a entrada reset do flip-flop SR (estado do SR vai para 0)

Para executar uma operação de escrita numa célula, deve-se acertar o valor de $input$, colocar o valor de $\overline{Read}/Write$ em 1 e, em seguida, fazer $select$ igual a 1.

No exemplo abaixo, uma memória com 4 palavras de 2 bits cada.



Unidade de controle

Maestro?

Circuito que é responsável por acertar os sinais de controle, para que o fluxo de dados e a execução das instruções ocorra corretamente.

- programa a ser executado: sequência de instruçõees armazenadas na memória
- instrução: código (da instrução) + [endereço]
- Exemplos de códigos de instruções (do HIPO; descrição completa em, por exemplo, https://www.ime.usp.br/~jstern/miscellanea/MaterialDidatico/hipo.htm)

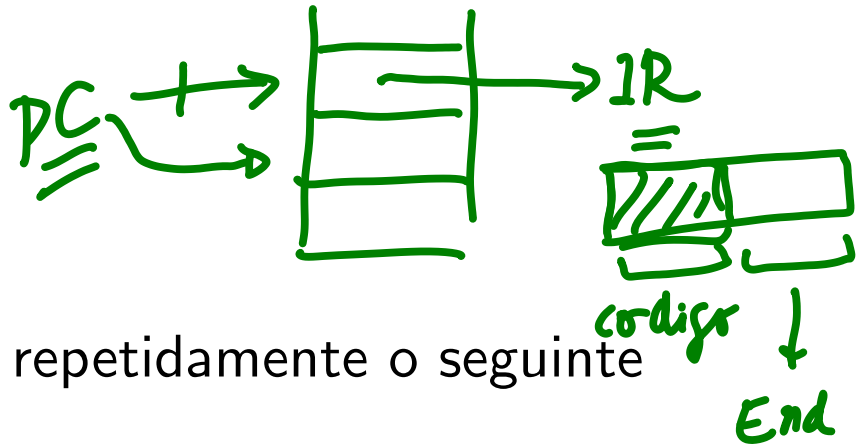
```
12: (CAE) Copie o conteúdo do acumulador no endereço EE.
21: (SOM) Some o conteúdo do endereço EE com o conteúdo do acumulador e guarde o resultado
no acumulador.
22: (SUB) Subtraia o conteúdo do endereço EE do conteúdo do acumulador e guarde o resultado
no acumulador.
70: (PAR) Pare a execução do programa. OBS.: Esta instrução deve ser executada para encerrar
a execução do programa.
```

Registradores especiais

PC – program counter → endereço da próxima instrução

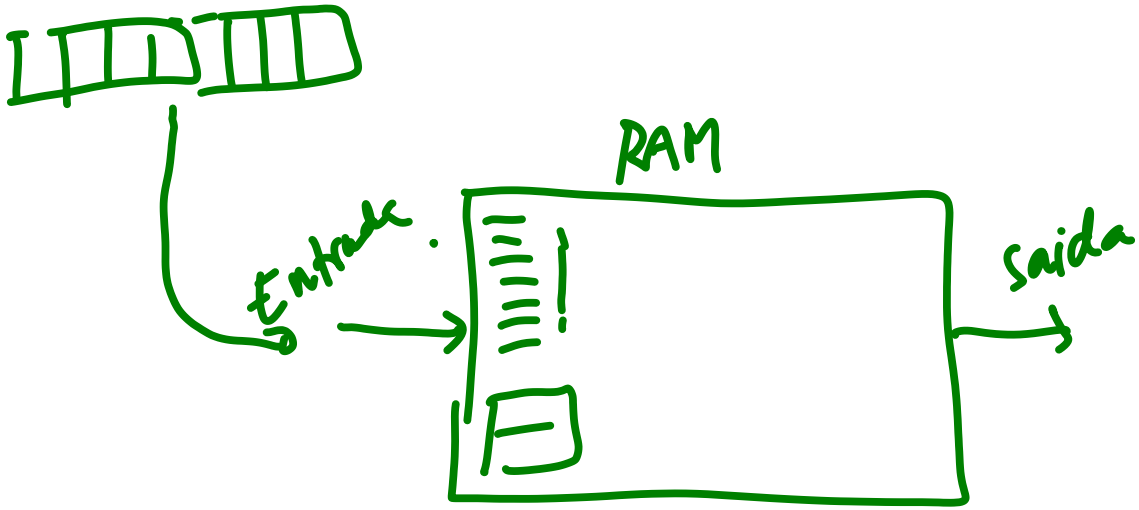
IR – instruction register → cópia da inst. a ser executada

ACC – accumulator → mem. auxiliar da ULA



A unidade de controle apenas executa repetidamente o seguinte ciclo:

1. Busca a próxima instrução a ser executada
2. Incrementar o PC
3. Decodifica a instrução a ser executada (prepara as entradas e os sinais de controle dos subcircuitos)
4. Executa a instrução


RAM
Entrada
Saída