

Driver serial para cMIPS

Integrantes:

Luigi Muller

Tarlison Sander

A Interface Serial cMIPS

- Professor. PhD. Roberto André Hexsel;
- “I always felt the need for some form of the processor for the students to play with”;
- É um modelo VHDL sintetizável para os clássicos 5 estágios do pipeline
 - ◆ Busca da instrução;
 - ◆ Decodificação
 - ◆ Execução
 - ◆ Memória
 - ◆ Write back



A Interface Serial cMIPS

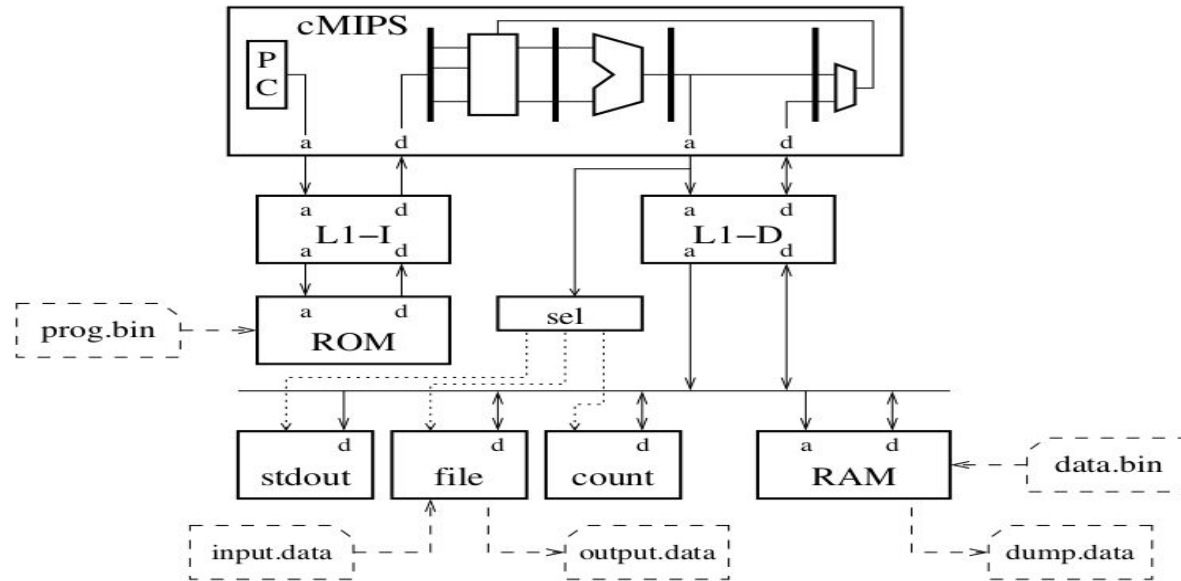


Figure 1: Block diagram of the “simple computer” in the testbench.

→ Ideia do funcionamento do seu ‘simple computer’

The diagram also shows the files needed for the model to run C/assembly programs. T

A Interface Serial cMIPS

- O driver é que irá interagir com a interface serial irá receber entradas para gerar o n-ésimo número de fibonnaci;
- O driver é dividido em duas partes:
 - ◆ Handler;
 - ◆ Conjunto de funções
- O handler funcionará no “meio do caminho” entre a UART e o código em C;
 - ◆ Possui nele a fila de recepção e a fila de transmissão;



Big Picture

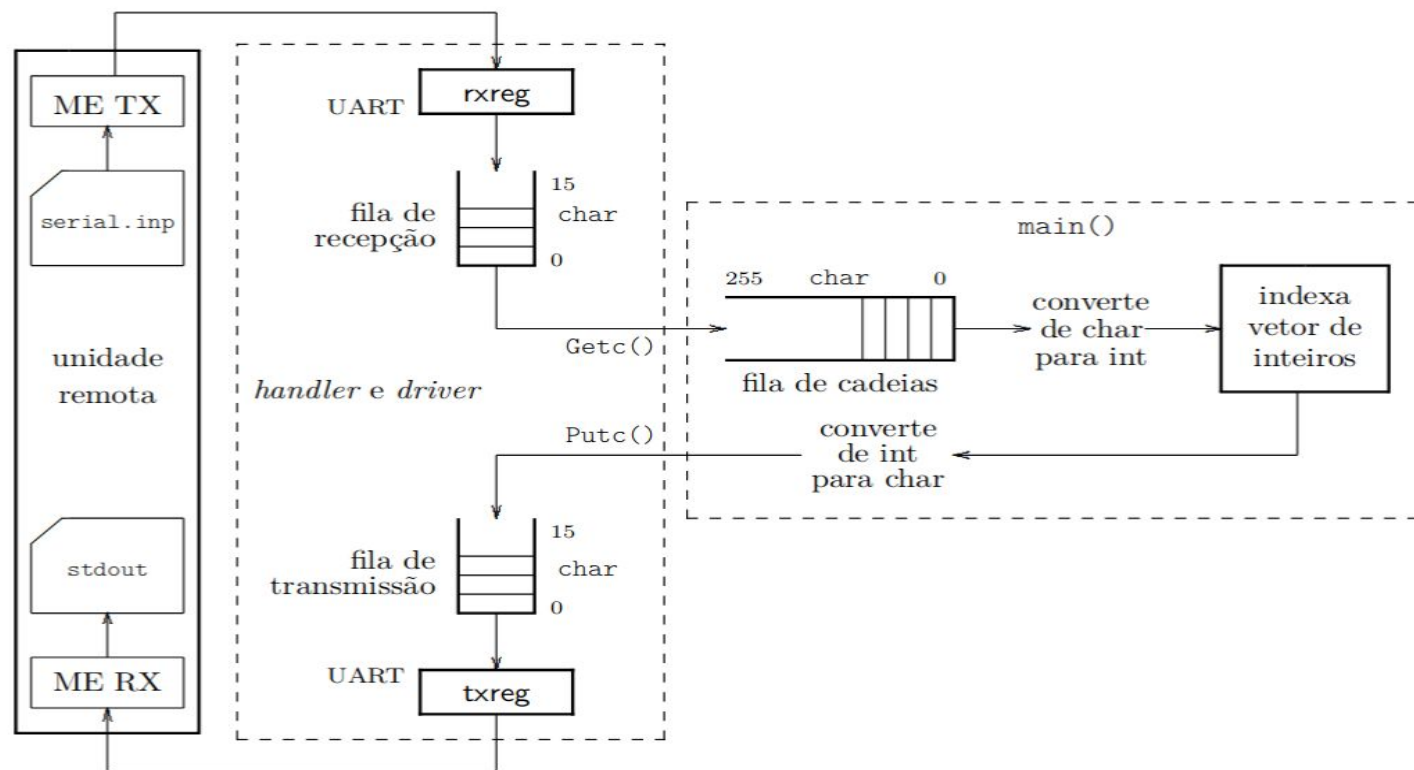
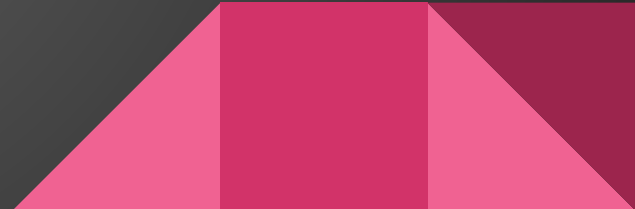


Figura 2: Fluxo de dados no *driver* e aplicação da UART.

A Interface Serial cMIPS

- O handler mantém dois contadores o nrx e o ntx;
 - ◆ nrx : indica o número de caracteres disponíveis na fila de recepção;
 - ◆ ntx: indica o número de espaços na fila de transmissão;
- O driver possui as funções:
 - ◆ proberx() e probetx();
 - ◆ iostat(); //leitura
 - ◆ ioctl(); //escrita
 - ◆ Getc(); //nrx
 - ◆ Putc(); //ntx



A Interface Serial cMIPS

→ Para o RX

```
ASM handlerUART.s
home ▸ tarlison ▸ Documentos ▸ Killer ▸ cMIPS ▸ tests ▸ ASM handlerUART.s
1  #-----
2  # interrupt handler for UART
3  -----
4  RX:
5      andi    $a0, $k1, UART_rx_irq
6      beq     $a0, $zero, TX
7
8      lui     $a0, %hi(Ud)
9      ori     $a0, $a0, %lo(Ud)
10
11     lw      $a1, 48($a0)
12
13     addiu   $k1, $zero, 16
14     slt     $k1, $a1, $k1          # se nrx >= 16 a fila esta cheia
15     beq     $k1, $zero, END
16
17     addiu   $a1, $a1, 1           # incrementa o nrx
18     sw      $a1, 48($a0)
19
20     lw      $a1, 20($a0)
21     nop
22     addiu   $k1, $a1, 1
23     andi    $k1, $k1, 15
24     sw      $k1, 20($a0)
25
26     add     $a0, $a1, $a0
27
28     lui     $a1, %hi(HW_uart_addr)
29     ori     $a1, $a1, %lo(HW_uart_addr)
30     lbu     $k1, 4($a1)
31     nop
32     sb      $k1, 0($a0)          # coloca os dados no fila RX
33
34     # lui $a0, %hi(x_IO_BASE_ADDR)
35     # ori $a0, $a0, %lo(x_IO_BASE_ADDR)
36     # sw $k1, 0($a0)
37
38     lw      $k1, 0($a1)
39     nop
40     j       RX
41     nop
42
```

A Interface Serial cMIPS

→ Para o TX

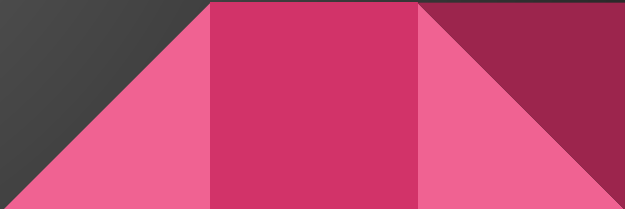
```
ASM handlerUART.s x
home ▸ tarlison ▸ Documentos ▸ Killer ▸ cMIPS ▸ tests ▸ ASM handlerUART.s

46 TX:
47     andi    $a0, $k1, UART_tx_irq
48     beq     $a0, $zero, END
49
50     lui     $a0, %hi(Ud)
51     ori     $a0, $a0, %lo(Ud)      # $a0 <- Ud
52
53     lw      $a1, 52($a0)           # Read ntx
54
55     # lui $k1, %hi(x_IO_BASE_ADDR)
56     # ori $k1, $k1, %lo(x_IO_BASE_ADDR)
57     # sw $a1, 0($k1)
58
59     addiu   $k1, $zero, 16
60     slt     $k1, $a1, $k1          # If ntx < 16 tem algo na fila
61     beq     $k1, $zero, END
62
63     addiu   $a1, $a1, 1           # Incrementa ntx
64     sw      $a1, 52($a0)
65
66     lw      $a1, 40($a0)
67     nop
68     addiu   $k1, $a1, 1
69     andi    $k1, $k1, 15
70     sw      $k1, 40($a0)
71
72     # addiu $a1, $a1, 24
73     add     $a0, $a1, $a0
74     lbu     $a1, 24($a0)
75
76     lui     $a0, %hi(HW_uart_addr)
77     ori     $a0, $a0, %lo(HW_uart_addr)
78     sb      $a1, 4($a0)          # Coloca o dado na UART
79
80     # lui $a0, %hi(x_IO_BASE_ADDR)
81     # ori $a0, $a0, %lo(x_IO_BASE_ADDR)
82     # sw $a1, 0($a0)
83
84     lw      $k1, 0($a0)
85     nop
86     j       RX
87     nop
88
89 END:
```


A Interface Serial cMIPS

→ No main:

```
◆ int main() {  
    uart = (void *)IO_UART_ADDR; // bottom of UART address range  
    int ictl = 0x1a; // 00011010: ign=0; intTX=1; intRX=1; speed=2;  
    ioctl(ictl);  
    initUd();  
    .....mais códigos aqui.....  
}
```



A Interface Serial cMIPS

→ Fibonacci usado no main:

```
int fibonacci(int entrada){
    if(entrada == 0 || entrada == 1){
        return 1;
    } else {
        int a = 1, b = 1;
        int c;
        for (int cont = 2; cont <= entrada; cont++){
            c = a + b;
            a = b;
            b = c;
        }
        return c;
    }
}
```

```
127
128 int proberx(){
129     return Ud.nrx;
130 }
131
132 int probetx(){
133     return Ud.ntx;
134 }
135
136 int iostat(){
137     return uart->cs.stat.s;
138 }
139
140 void ioctl(int ictl){
141     Tcontrol ctrl;
142     ctrl.speed = ictl & 0x7;
143     ctrl.intRX = (ictl>>3) & 0x1;
144     ctrl.intTX = (ictl>>4) & 0x1;
145     ctrl.ign    = (ictl>>5) & 0x7;
146     uart->cs.ctl = ctrl;
147 }
148
149 char getc(){
150     char c = EOF;
151     if(proberx() > 0){
152         disableInterr();
153         c = Ud.rx_q[Ud.rx_hd];
154         Ud.rx_hd = (Ud.rx_hd+1)%16;
155         Ud.nrx--;
156         enableInterr();
157     }
158     return c;
159 }
```

A Interface Serial cMIPS

```
160
161 void Putc(char c){
162     if(probetx() > 0){
163         disableInterr();
164         Ud.tx_q[Ud.tx_tl] = c;
165         Ud.tx_tl = (Ud.tx_tl+1)%16;
166         Ud.ntx--;
167         enableInterr();
168     }else{
169         writeFirstChar();
170         Putc(c);
171     }
172 }
173
```

Obrigado pela atenção!



Talk is cheap. Show me the code.

— *Linus Torvalds* —