

INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

SSC0511 Organização de Computadores Digitais

Prof. Dr. Eduardo do Valle Simoes

Grupo 7 - Ellian Carlos (11846324), Giovanna Fardini (10260671), Thales Damasceno (11816150) e Vinicius Baca (10788589).

Link do repositório: <https://github.com/gifardini/SSC0511-2020>

Parte 1 – Nova instrução para o processador ICMC: EXCHANGE

Para a primeira parte da avaliação da disciplina de organização de computadores digitais o grupo fez a criação de uma nova instrução para o processador ICMC. O grupo optou por criar a instrução *Exchange* (EXCHG), presente em outros processadores e que pode ser bastante útil para a troca de dados entre memória e registrador. A instrução XCHG recebe como argumento um registrador e um endereço de memória, respectivamente, e faz a troca do conteúdo dos dois, isto é, o valor do registrador será escrito na memória e o dado da memória será passado para o registrador. Esta instrução é realizada em 3 bordas de *clock*, sendo um ciclo de decodificação e dois ciclos de execução.

Para que a nova instrução pudesse ser implementada corretamente foi preciso adicionar um novo registrador na arquitetura como mostrado na figura abaixo, o novo registrador, chamado de *memory buffer register* (MBR), é utilizado como auxiliar para se fazer a troca entre os dados do registrador e da memória. As linhas em amarelo representam o trajeto dos dados durante o funcionamento da instrução.


```
testaCPU.asm
3
4 ; 4 Perguntas ao implementar as instrucoes:
5 ; 1) O Que preciso fazer para esta instrucao?
6 ; 2) Onde Comeca: Pegar gcc simple_simulator.c -O3
   -march=native -o simulador -Wall -lm -lcurses o que tem
   que fazer e ir voltando ate' chegar em um registrador
   (ie. PC)
7 ; 3) Qual e' a Sequencia de Operacoes: Descrever todos os
   comandos que tem que dar nos ciclos de Dec e Exec
8 ; 4) Ja' terminou??? Cumpru o que tinha que fazer??? O
   PC esta' pronto para a proxima instrucao (cuidado com
   Load, Loadn, Store, Jmp, Call)
9
10 ; Teste do xchg
11 loadn r1, #'C'
12 loadn r0, #1
13 store Dado, r1
14 loadn r2, #'Y'
15 outchar r2, r0
16 xchg r2, Dado
17 outchar r2, r0
18
19
20 Fim:
21 halt
22
23
```

```
> ./mont/montador testaCPU.asm CPURAM.mif
Montador v.0.0
Mensagem (0): Encontrando labels...
Mensagem (20): Label "Fim" em 0xc.
Mensagem (23): Label "Dado" em 0xd.
Mensagem (0): Montando codigo...
Mensagem (0): Inicializando buffer de saida...
Mensagem (24): Descarregando buffer de saida...
Mensagem (24): Concluido.
> ./simulador
Rodando...
YC
```

O primeiro código armazena a letra B na memória e A no registrador 2 e imprime o dado armazenado no registrador antes e depois de realizar a troca. Dessa forma, na primeira impressão o resultado é a letra A e na segunda a letra B, conforme demonstrado. O segundo código aplica o mesmo processo, porém utilizando as letras C e Y e após a troca serão impressas Y e C.