Instituto Federal do Espírito Santo – Campus Serra

Gabriel Couceiro Figueiredo
Gabriel Rodrigues Barbosa

Instruções de uso e observações

Serra
2017

Instituto Federal do Espírito Santo – Campus Serra

Gabriel Couceiro Figueiredo
Gabriel Rodrigues Barbosa

Instruções de uso e observações

Relatório apresentado à disciplina de Arquitetura de Computadores, do curso de Engenharia de Controle e Automação, do Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo, como requisito parcial para avaliação na referida disciplina.
Orientador: Prof. Rafael Emerick

Serra
2017

Endereço do projeto no GitHub: https://github.com/gabrielrb1/Trabalho-2---ArqComp

1) Em relação ao programa da PG

O primeiro programa utilizado no Visual é o de nome “progressao_aritmetica”. Neste programa, foi feito o uso de branches para termos os loops. Entretanto, tal solução foi abandonada, uma vez que as ferramentas que tínhamos para fazer a tradução do código para linguagem de máquina não fazem a tradução de branches. Com isso, foi feito um segundo algoritmo, utilizando como solução aos branches, a manipulação do registrador 15, referente ao PC. Tal programa está no repositório com o nome “pq2”. Infelizmente, nenhum dos dois algortimos foi implementado com sucesso.

2) Posição da memória

Conforme solicitado pelo professor na especificação do projeto, o endereço de memória do primeiro registro de nosso algoritmo deveria ser o de número 1000. Entretanto, conforme exposto durante a apresentação do projeto, a RAM criada no processador monociclo só suporta 256 endereços. Para solucionar tal conflito, haviam duas opções: alterar no código do processador o tamanho da RAM ou alterar no algoritmo da PG o endereço inicial para um valor que fosse suportado pela memória. A alternativa escolhida foi a segunda.

3) Implementação das instruções CMP e TST

Para implementar as duas instruções indicadas acima, foram necessárias alterações no plano do controle, conforme indicado pela imagem abaixo.

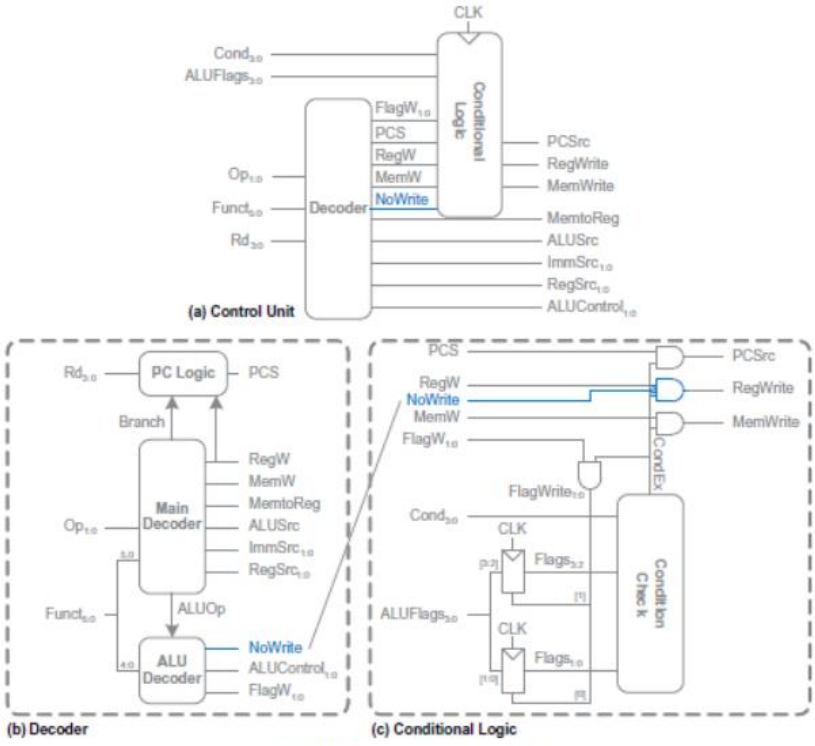


Figure 7.16 Controller modification for CMP

As duas instruções implementadas são realizam as mesmas operações que SUB e AND, respectivamente, com a diferença de que não escrevem o valor obtido em algum registrador, apenas atualizam as flags.
Também foi necessário realizar nas opções de controle da ALU, associando cada uma das novas instruções com sua instrução equivalente.
Para validar o funcionamento de cada uma, foram usados algoritmos de teste, conforme descrito abaixo.

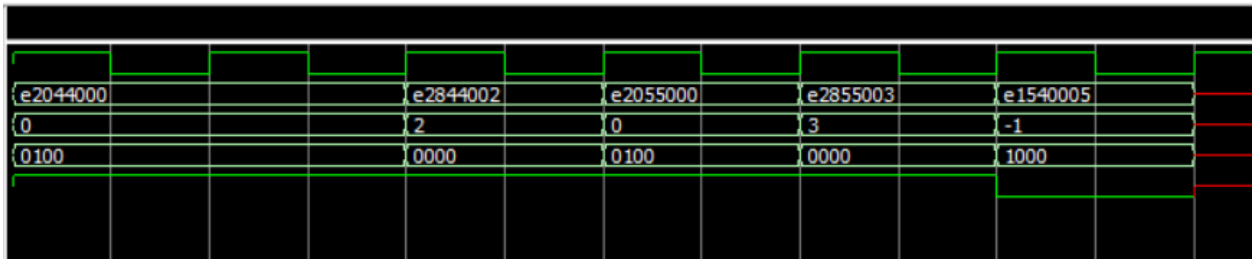
```
and r4, r4,#0
add r4, r4, #2
and r5, r5, #0
add r5, r5, #3
cmp r4,r5
```

Código de teste usado

```
E2044000
E2844002
E2055000
E2855003
E1540005
```

Linguagem de máquina equivalente

Após compilar e exportar as variáveis para o ambiente “wave”, temos a imagem abaixo.



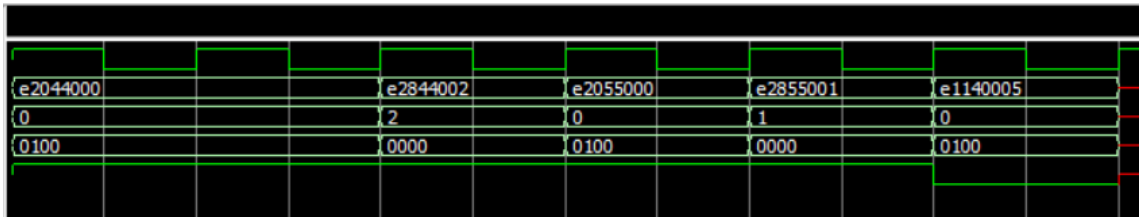
A primeira linha indica o clock, a segunda indica a instrução sendo lida, a terceira é o valor que o registrador recebeu no momento (detalhe para o resultado da CMP na última instrução, -1), a quarta são as flags e por último, temos o NoWrite.
Em seguida, o procedimento foi repetido para validar o TST.

```
and r4, r4,#0
add r4, r4, #2
and r5, r5, #0
add r5, r5, #1
tst r4,r5
```

Código de teste

E2044000
E2844002
E2055000
E2855001
E1140005

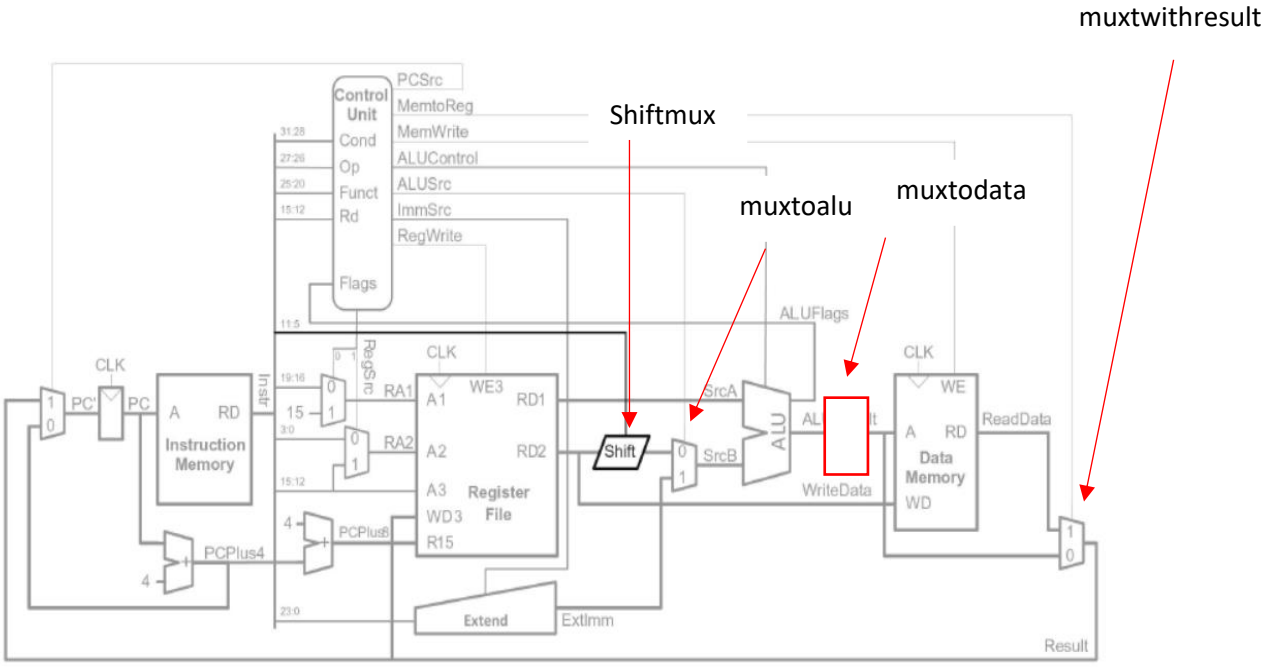
Linguagem de máquina



Por ser uma operação binária do tipo AND entre 2 (0010) e 1 (0001), o resultado foi zero.

4) Implementação da instrução LSL

Para fazer a implementação da instrução LSL, foi necessário fazer alterações no plano de controle, ao adicionar uma adaptação no decoder para decodificar tal instrução; e no datapath, ao adicionar dois mux e adaptar os dois existentes.
A primeira alteração, assim como foi feito para TST e CMP, foi adaptar a ALU para reconhecer o LSL. Como tal instrução não possui nenhuma outra semelhante, foi alterado o algoritmo do processador de forma que a opção “default” da estrutura switch/case direcionasse para LSL. Outra solução seria aumentar a quantidade de bits de controle da ALU, o que permitiria a inclusão de novas instruções.
Em relação ao mux adicionados no datapath, o primeiro teve como função fazer a operação de rotação propriamente dita e o segundo teve como função selecionar ou não o que foi calculado na ALU.

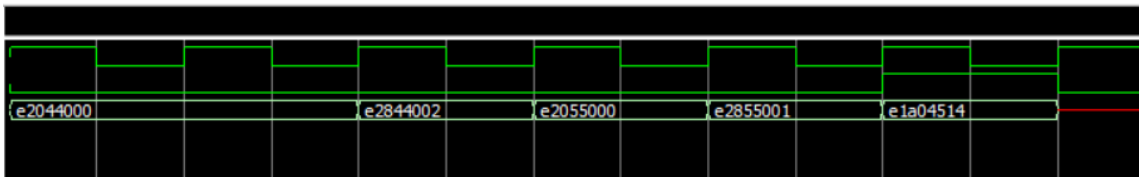


A exemplo das outras instruções, também utilizamos um algoritmo de teste.

and r4, r4,#0
add r4, r4, #2
and r5, r5, #0
add r5, r5, #1
lsl r4,r4,r5
Algoritmo usado

E2044000
E2844002
E2055000
E2855001
E1A04514
Linguagem de máquina

E abaixo temos o resultado da simulação. Não foi exibido o resultado da operação, pois na simulação foi apresentado um valor incorreto. Entretanto, é possível veificar que a instrução foi reconhecida pelo processador. Isso pode ser notado pois na momento em que o LSL está sendo executado, o “Shift” é ativado.



A primeira linha indica o clock, a segunda indica o Shift e a terceira indica a instrução sendo executada.

5) Arquivos “memfile”

Os arquivos com nome “memfile” representam o programa emulado no Visual na forma de linguagem de máquina.