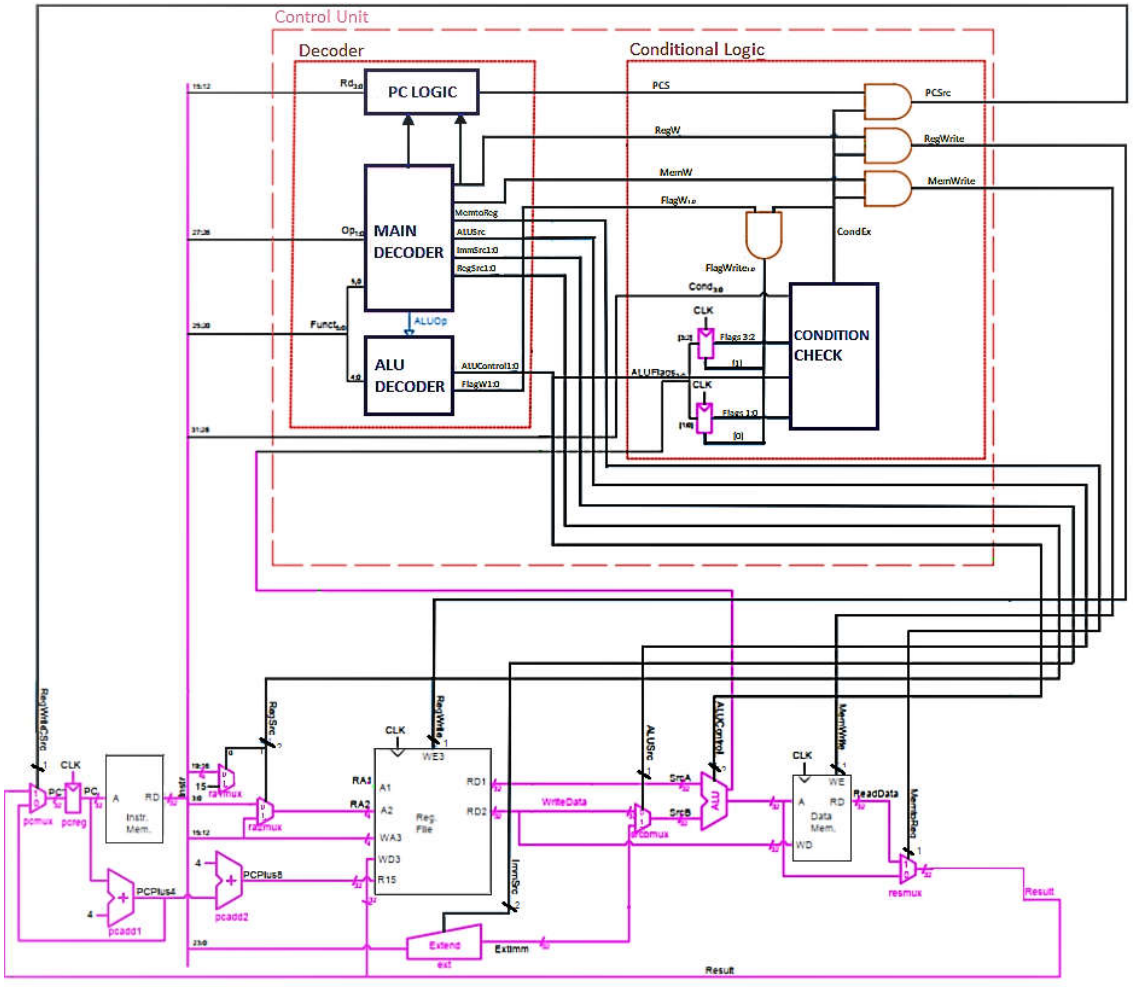
TRABALHO PRÁTICO 2018/2
EXTENSÃO DE INSTRUÇÃO DE PROCESSADOR ARM
MONOCICLO.
Alunos : Bruno;Ravena;Débora

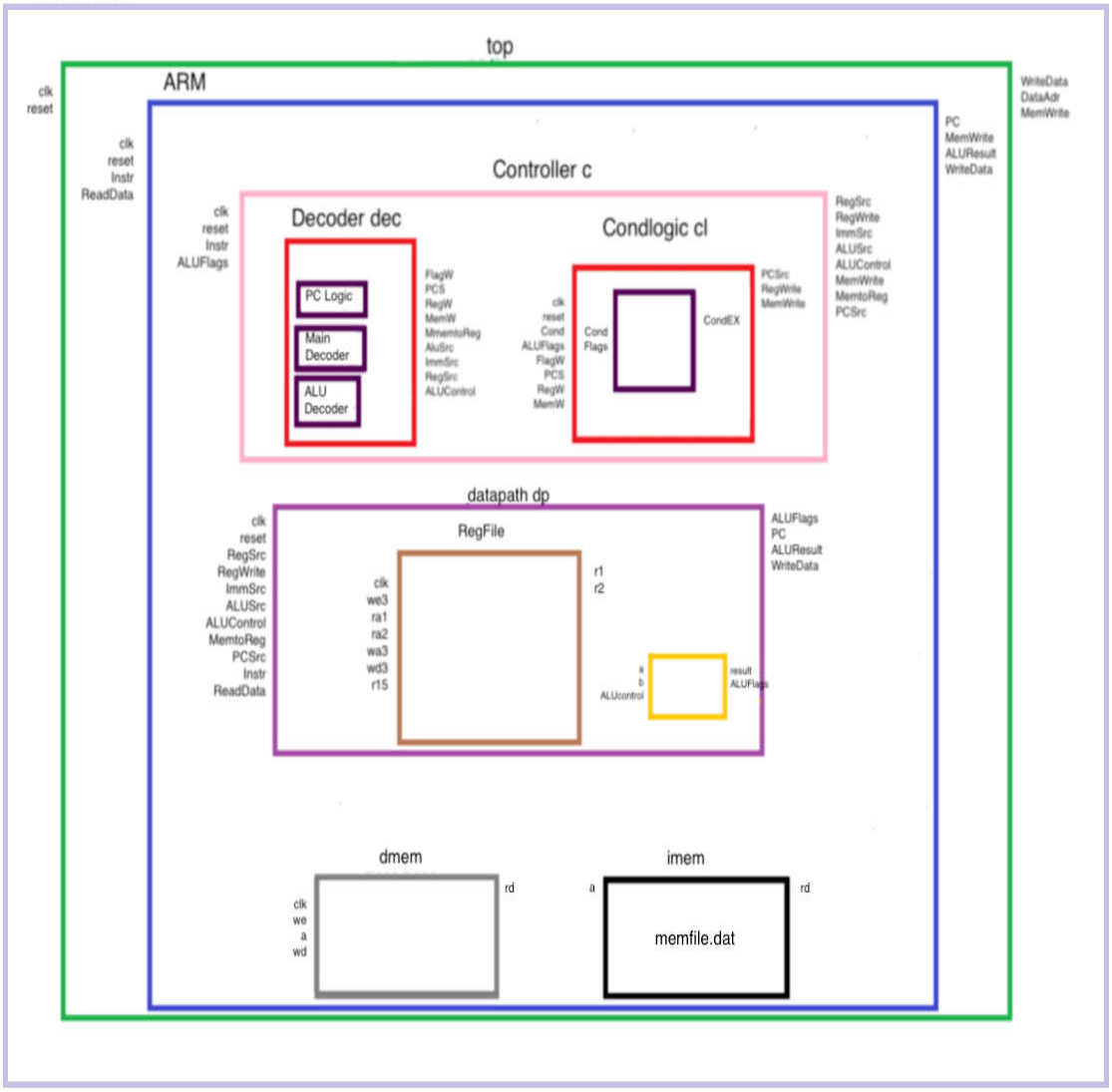
Breve roteiro da execução do trabalho com principais modificações implementadas

Parte 1 (3 pts) – Testando o processador ARM monociclo

1 - Desenhe o esquemático correspondente ao HDL em SystemVerilog do processador ARM. Ilustre a hierarquia do testbench e os módulos internos ao DUT (Device Under Test). Apresente no esquemático as estruturas de conexão de forma simplificada (para fios em paralelo, trace apenas 1 fio e ilustre o número de fios que compõem tal barramento simples)



testbench


top
ARM
clk
reset
WriteData
DataAdr
MemWrite
clk
reset
Instr
ReadData
PC
MemWrite
ALUResult
WriteData
Controller c
clk
reset
Instr
ALUFlags
Decoder dec
PC Logic
Main Decoder
ALU Decoder
FlagW
PCS
RegW
MemW
MmemtoReg
AluSrc
ImmSrc
RegSrc
ALUControl
Condlogic cl
clk
reset
Cond
ALUFlags
FlagW
PCS
RegW
MemW
Cond Flags
CondEX
PCSrc
RegWrite
MemWrite
RegSrc
RegWrite
ImmSrc
ALUSrc
ALUControl
MemWrite
MemtoReg
PCSrc
datapath dp
clk
reset
RegSrc
RegWrite
ImmSrc
ALUSrc
ALUControl
MemtoReg
PCSrc
Instr
ReadData
RegFile
clk
we3
ra1
ra2
wa3
wd3
r15
r1
r2
a
b
ALUcontrol
result
ALUFlags
ALUFlags
PC
ALUResult
WriteData
dmem
clk
we
a
wd
rd
imem
a
rd
memfile.dat

2 - Preencha o gráfico na Tabela 1 no final com suas previsões da execução do código memfile.s do repositório fornecido. Que endereço escreverá a instrução final do STR e qual valor ela escreverá? Simule seu processador com o ModelSim. Adicione todos os sinais da Tabela 1 à sua janela de ondas. Execute a simulação. Se tudo correr bem, o testbench imprimirá “Simulação bem-sucedida”. Observe as formas de onda e verifique se elas correspondem às suas previsões na Tabela 1.

Ciclo	Reset	PC	Instrução (Montagem / Máquina)	SrcA	SrcB	Branch	AluResult	Flags3:0	CondEx	WriteData	MemWrite	ReadData
1	1	0x00000000	E04F000F	0x00000008	0x00000008	0	0x00000000	0110	1	0x00000008	0	0xxxxxxxxx
2	1	0x00000000	E04F000F	0x00000008	0x00000008	0	0x00000000	0110	1	0x00000008	0	0xxxxxxxxx
3	1	0x00000000	E04F000F	0x00000008	0x00000008	0	0x00000000	0110	1	0x00000008	0	0xxxxxxxxx
4	0	0x00000004	E2802005	0x00000000	0x00000005	0	0x00000005	0000	1	0xxxxxxxxx	0	0xxxxxxxxx
5	0	0x00000008	E280300C	0x00000000	0x0000000c	0	0x0000000c	0000	1	0xxxxxxxxx	0	0xxxxxxxxx
6	0	0x0000000c	E2437009	0x0000000c	0x00000009	0	0x00000003	0010	1	0xxxxxxxxx	0	0xxxxxxxxx
7	0	0x00000010	E1874002	0x00000003	0x00000005	0	0x00000007	0000	1	0x00000005	0	0xxxxxxxxx
8	0	0x00000014	E0035004	0x0000000c	0x00000007	0	0x00000004	0000	1	0x00000007	0	0xxxxxxxxx
9	0	0x00000018	E0855004	0x00000004	0x00000007	0	0x0000000b	0000	1	0x00000007	0	0xxxxxxxxx
10	0	0x0000001c	E0558007	0x0000000b	0x00000003	0	0x00000008	0010	1	0x00000003	0	0xxxxxxxxx
11	0	0x00000020	0A00000C	0x00000028	0x00000030	1	0x00000058	0000	0	0xxxxxxxxx	0	0xxxxxxxxx
12	0	0x00000024	E0538004	0x0000000c	0x00000007	0	0x00000005	0010	1	0x00000007	0	0xxxxxxxxx
13	0	0x00000028	AA000000	0x00000030	0x00000000	1	0x00000030	0000	1	0x00000000	0	0xxxxxxxxx
14	0	0x00000030	E0578002	0x00000003	0x00000005	0	0xfffffffe	1000	1	0x00000005	0	0xxxxxxxxx
15	0	0x00000034	B2857001	0x0000000b	0x00000001	0	0x0000000c	0000	1	0xxxxxxxxx	0	0xxxxxxxxx
16	0	0x00000038	E0477002	0x0000000c	0x00000005	0	0x00000007	0010	1	0x00000005	0	0xxxxxxxxx
17	**0**	**0x0000003c**	**E5837054**	**0x0000000c**	**0x00000007**	**0**	**0x00000060**	**0000**	**1**	**0x00000007**	**1**	**0xxxxxxxxx**
18	0	0x00000040	E5902060	0x00000000	0x00000000	0	0x00000060	0000	1	0x00000000	0	0x00000007
19	0	0x00000048	E08ff000	0x0000004c	0x00000000	0	0x0000004c	0000	1	0x00000000	0	0xxxxxxxxx

A instrução final do STR e o valor que ela escreverá está no ciclo 17, com o endereço 0x3c e valor 0x60.

Parte 2 (7 pts): - Modificando o processador ARM monociclo

1 - Estenda as instruções em nível de microarquitetura do processador ARM monociclo, com código HDL disponível em [1], de forma a habilitá-lo a processar as instruções TST, CMP, LSL, MOV, EOR e LDRB.

ALU Decoder

Op	cmd	S	Type	ALUControl	FlagW	ShiftIt
00	1010	1	CMP	001	11	0
00	1000	1	TST	010	10	0
00	0001	0	EOR	100	00-11	0
00	1101	0	LSL	-	00	1
00	1101	0	MOV	-	00	1
01	-	-	LDRB	-	00	0

2 - Para validar, estenda o código de testes utilizado na parte , gere o novo código memfile2.s e memfile2.dat, com os códigos correspondentes de montagem e de máquina. Por fim, atualize a tabela 1 com os dados obtidos dessa nova simulação com as novas instruções suportadas.

```
// memfile2.s
// david_harris@hmc.edu and sarah.harris@unlv.edu 20 Jan 2014
// Test ARM processor
// ADD, SUB, AND, ORR, LDR, STR, B
// TST, LSL, CMN, ADC
// If successful, it should write the value 7 to address 100

//  MOV
//  CMP
// TST
//  LSL
//  EOR
// LDRB

MAIN    MOV R0, #3                      ; R0 = 3
        EOR R0, R0, #3                  ; Invert bits zero and one of R0
        SUB R0, R15, R15  ; R0 = 0
        ADD R2, R0, #5    ; R2 = 5
        CMP R2, R0                      ; R2 > R0, flags Z == 0, N == 1
        ADD R3, R0, #12   ; R3 = 12
        SUB R7, R3, #9    ; R7 = 3
        TST R7, R2        ; set N and Z flags to 1 and 0
        ORR R4, R7, R2    ; R4 = 3 OR 5 = 7
    AND R5, R3, R4        ; R5 = 12 AND 7 = 4
        ADD R5, R5, R4    ; R5 = 4 + 7 = 11
    SUBS R8, R5, R7       ; R8 <= 11 - 3 = 8, set Flags
    BEQ END               ; shouldn't be taken
    SUBS R8, R3, R4       ; R8 = 12 - 7  = 5
   // BGE AROUND          ; should be taken
        ADD R5, R0, #0    ; should be skipped
//AROUND
        SUBS R8, R7, R2   ; R8 = 3 - 5 = -2, set Flags
    ADDLT R7, R5, #1      ; R7 = 11 + 1 = 12
    SUB R7, R7, R2        ; R7 = 12 - 5 = 7
        STR R7, [R3, #84] ; mem[12+84] = 7
        LDR R2, [R0, #96] ; R2 = mem[96] = 7
        LSL R3, R2, #3                  ; R2 = R2 << 3
        ADD R15, R15, R0  ; PC <- PC + 8 (skips next)
        ADD R2, R0, #14   ; shouldn't happen
        //B END           ; always taken
        ADD R2, R0, #13   ; shouldn't happen
        ADD R2, R0, #10                 ; shouldn't happen
        LDRB R1, [R2, #1] ; R1 = mem[R2,1]
        STR R2, [R0, #100] ; mem[100] = 7

// memfile2.s
E2200003
E3A0000C
E04F000F
E2802005
E1520000
E280300C
E2437009
E1170002
E1874002
```

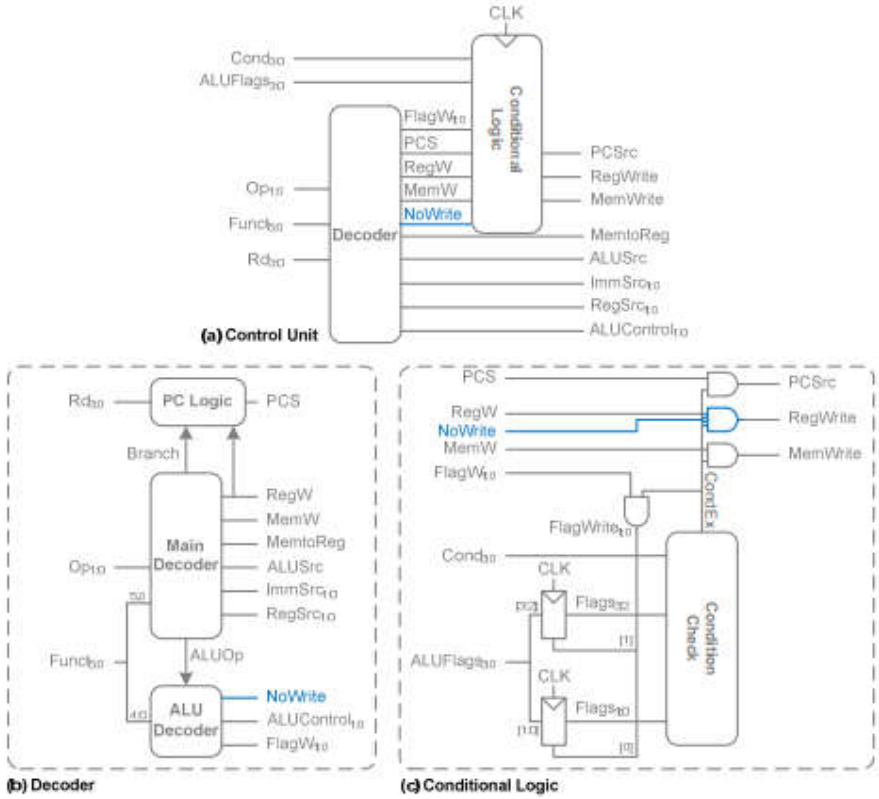
E0035004
E0855004
E0558007
0A00000C
E0538004
AA000000
E2805000
E0578002
B2857001
E0477002
E5837054
E5902060
E1A02182
E08FF000
E280200E
EA000001
E280200D
E280200A
E5D21001
E5802064

TABELA ATUALIZADA COM AS NOVAS INSTRUÇÕES ACIMA

Ciclo	Reset	PC	Instrução (Montagem / Máquina)	SrcA	SrcB	Branch	AluResult	Flags3:0	CondEx	WriteData	MemWrite	ReadData
1	1	0x00000000	E2200003	0xxxxxxxxx	0x00000003	0	0xxxxxxxxx	0000	1	0xxxxxxxxx	0	0xxxxxxxxx
2	1	0x00000000	E2200003	0xxxxxxxxx	0x00000003	0	0xxxxxxxxx	0000	1	0xxxxxxxxx	0	0xxxxxxxxx
3	1	0x00000000	E2200003	0xxxxxxxxx	0x00000003	0	0xxxxxxxxx	0000	1	0xxxxxxxxx	0	0xxxxxxxxx
4	0	0x00000004	E3A0000C	0x00000000	0x00000005	0	0x0000000c	0000	1	0xxxxxxxxx	0	0xxxxxxxxx
5	0	0x00000008	E04F000F	0x00000010	0x00000010	0	0x00000000	0000	1	0x0000000c	0	0xxxxxxxxx
6	0	0x0000000c	E2802005	0x00000000	0x00000005	0	0x00000005	0000	1	0xxxxxxxxx	0	0x000000xx
7	0	0x00000010	E1520000	0x00000005	0x00000000	0	0x00000005	0000	1	0x00000000	0	0x000000xx
8	0	0x00000014	E280300C	0x00000000	0x0000000c	0	0x0000000c	0010	1	0xxxxxxxxx	0	0xxxxxxxxx
9	0	0x00000018	E2437009	0x0000000c	0x00000009	0	0x00000003	0010	1	0xxxxxxxxx	0	0x000000xx
10	0	0x0000001c	E1170002	0x00000003	0x00000005	0	0x00000001	0010	1	0x00000000	0	0x000000xx
11	0	0x00000020	E1874002	0x00000003	0x00000005	0	0x00000007	0010	1	0xxxxxxxxx	0	0x000000xx
12	0	0x00000024	E0035004	0x0000000c	0x00000007	0	0x00000004	0010	1	0xxxxxxxxx	0	0xxxxxxxxx
13	0	0x00000028	E0855004	0x00000004	0x00000007	0	0x0000000b	0010	1	0x00000004	0	0x000000xx
14	0	0x0000002c	E0558007	0x0000000b	0x00000003	0	0x00000008	0010	1	0xxxxxxxxx	0	0xxxxxxxxx
15	0	0x00000030	0A00000C	0x00000038	0x00000030	1	0x00000068	0010	0	0x00000000	0	0xxxxxxxxx
16	0	0x00000034	E0538004	0x0000000c	0x00000007	0	0x00000005	0010	1	0x00000008	0	0x000000xx
17	0	0x00000038	AA000000	0x00000040	0x00000000	0	0x00000040	0010	1	0x00000000	0	0xxxxxxxxx
19	0	0x00000040	**E0578002**	0x00000003	0x00000005	0	0xfffffffe	0010	1	0x00000005	0	0x000000xx
20	0	0x00000044	B2857001	0x0000000b	0x00000001	0	0x0000000c	1000	1	0x00000003	0	0xxxxxxxxx
21	0	0x00000048	E0477002	0x0000000c	0x00000005	0	0x00000007	1000	1	0x0000000c	0	0x000000xx
22	0	0x0000004c	E5837054	0x0000000c	0x00000054	0	0x00000060	1000	1	0x00000007	1	0xxxxxxxxx
23	0	0x00000050	E5902060	0x00000000	0x00000060	0	0x00000060	1000	1	0x00000005	0	0x00000007
24	0	0x00000054	E1A02182	0x00000000	0x0000000e	0	0x0000000e	1000	1	0x00000007	0	0x000000xx
25	0	0x00000058	E08FF000	0x00000060	0x00000000	1	0x00000060	1000	1	0x00000060	0	0x00000007
27	0	0x00000060	EA000001	0x00000068	0x00000004	0	0x0000006c	1000	1	0x00000000	0	0xxxxxxxxx
30	0	0x0000006c	E5D21001	0x0000000e	0x00000001	0	0x0000000f	1000	1	0xxxxxxxxx	0	0x000000xx
31	0	0x00000070	E5802064	0x00000000	0x00000064	0	0x00000064	1000	1	0x0000000e	1	0xxxxxxxxx

ALU Decoder truth table

ALUOp	*Funct$_{4:1}$ (cmd)*	*Funct$_0$ (S)*	Notes	*ALUControl$_{1:0}$*	*FlagW$_{1:0}$*	*NoWrite*
0	X	X	Not DP	00	00	0
1	0100	0	ADD	00	00	0
		1			11	0
	0010	0	SUB	01	00	0
		1			11	0
	0000	0	AND	10	00	0
		1			10	0
	1100	0	ORR	11	00	0
		1			10	0
	1000	1	TST	10	10	1



- Função Nowrite : Garante que as instruções CMP e TST não escreva no registrador Reg.File.

- TST

Instrução similar ao AND, implementada na ULA, porém, não grava em registrador e atualiza flags.

- CMP

Operação similar ao SUB, que seta flags, mas não atualiza registradores graças ao nowrite;

- LSL

Função do LSL : Acrescentado um bloco de deslocamento - shifter - que tem a função de deslocar os bit para a esquerda. O valor do deslocamento pode ser imediato ou rm.

- LDRB

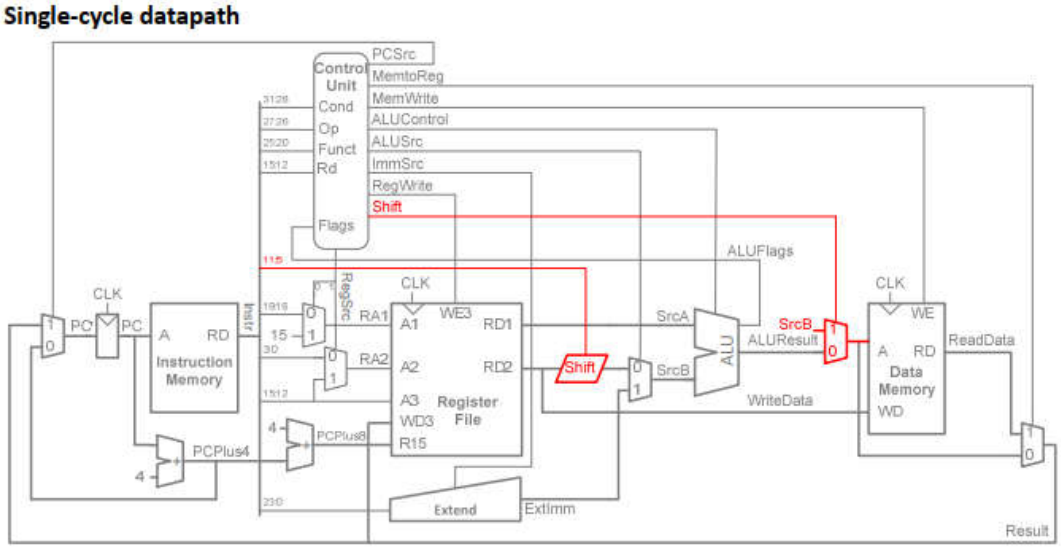
Foi modificada a função LDR para tratar a instrução Byte a Byte caso o bit “B”, Implementado como MenByte, seja “1” ou opere normalmente caso “0”;

- MOV

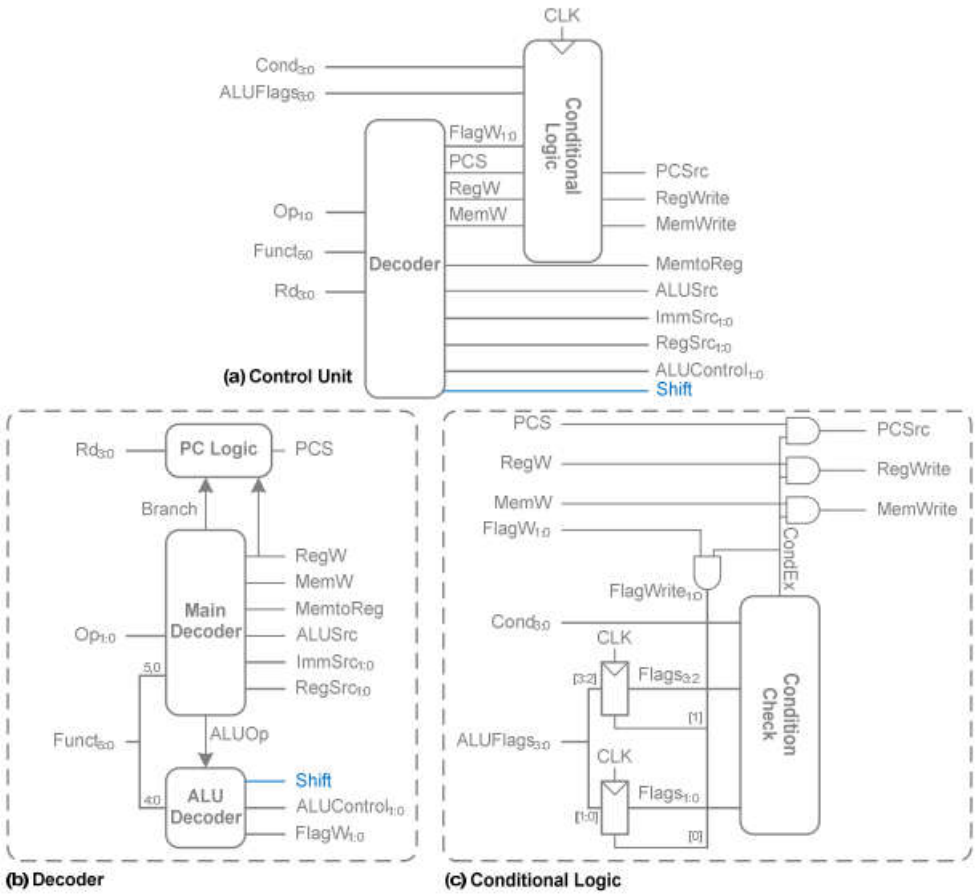
Foi necessário acrescentar mais um multiplexador de modo a não correr o risco de se utilizar ou modificar valores de R0 indevidamente;

- EOR

Para implementar o EOR, foi necessário aumentar a ALUcontrol em um bit para ser possível selecionar uma operação a mais.



Control unit



ALU Decoder truth table

ALUOp	*Funct*$_{4:1}$ *(cmd)*	*Funct*$_0$ *(S)*	Notes	*ALUControl*$_{1:0}$	*FlagW*$_{1:0}$	*Shift*
0	X	X	Not DP	00	00	0
1	0100	0	ADD	00	00	0
		1			11	0
	0010	0	SUB	01	00	0
		1			11	0
	0000	0	AND	10	00	0
		1			10	0
	1100	0	ORR	11	00	0
		1			10	0
	1101	0	LSL	XX	00	1
		1			10	1

Instrução	ALUcontrol	flag
ADD	000	-
SUB	001	-
AND	010	-
ORR	011	-
EOR	100	-
TST	010	N,Z,C
CMP	001	N,Z

Test ARM assembly code:

```
; If successful, it should write the value 2 to address 20

MAIN
  SUB R3, PC, PC        ; R3 = 0
  ADD R3, R3, #1        ; R3 = 0x1
  LSL R3, R3, #30       ; R3 = 0x80000000
  ADD R4, R3, #1        ; R4 = 0x80000001
  CMN R3, R4            ; set flags according to R3+R4: NZCV=0011
  ADC R3, R3, #5        ; R3 = 0x80000006
  TST R3, R4            ; set NZ flags according to R3&R4: NZCV=1011
  LSL R3, R3, #1        ; R3 = 0x0000000c
  LSL R4, R4, #1        ; R4 = 0x00000002
  STRVC R4, [R3, #4]    ; mem[16]<=0x2 if V=0:
                        ; shouldn't happen
  STRVS R4, [R3, #8]    ; mem[20]<=0x2 if V=1: should happen


; E04F300F  SUB       R3,PC,PC
; E2833001  ADD       R3,R3,#0x00000001
; E1A03F83  LSL       R3,R3,#31
; E2834001  ADD       R4,R3,#0x00000001
; E1730004  CMN       R3,R4
; E2A33005  ADC       R3,R3,#0x00000005
; E1130004  TST       R3,R4
; E1A03083  LSL       R3,R3,#1
; E1A04084  LSL       R4,R4,#1
; 75834004  STRVC     R4,[R3,#0x0004]
; 65834008  STRVS     R4,[R3,#0x0008]
```

ex7.9_memfile.dat

```
E04F300F
E2833001
E1A03F83
E2834001
E1730004
E2A33005
E1130004
E1A03083
```

```
E1A04084
75834004
65834008
```

- CMP

Extended Functionality: CMP

ALUOp	$Funct_{4:1}$ (*cmd*)	$Funct_0$ (*S*)	Type	$ALUControl_{1:0}$	$FlagW_{1:0}$	NoWrite
0	X	X	Not DP	00	00	**0**
1	0100	0	ADD	00	00	**0**
		1			11	**0**
	0010	0	SUB	01	00	**0**
		1			11	**0**
	0000	0	AND	10	00	**0**
		1			10	**0**
	1100	0	ORR	11	00	**0**
		1			10	**0**
	1010	**1**	**CMP**	**01**	**11**	**1**

Extended Functionality: CMP