Digital Design and Computer Architecture (E85)
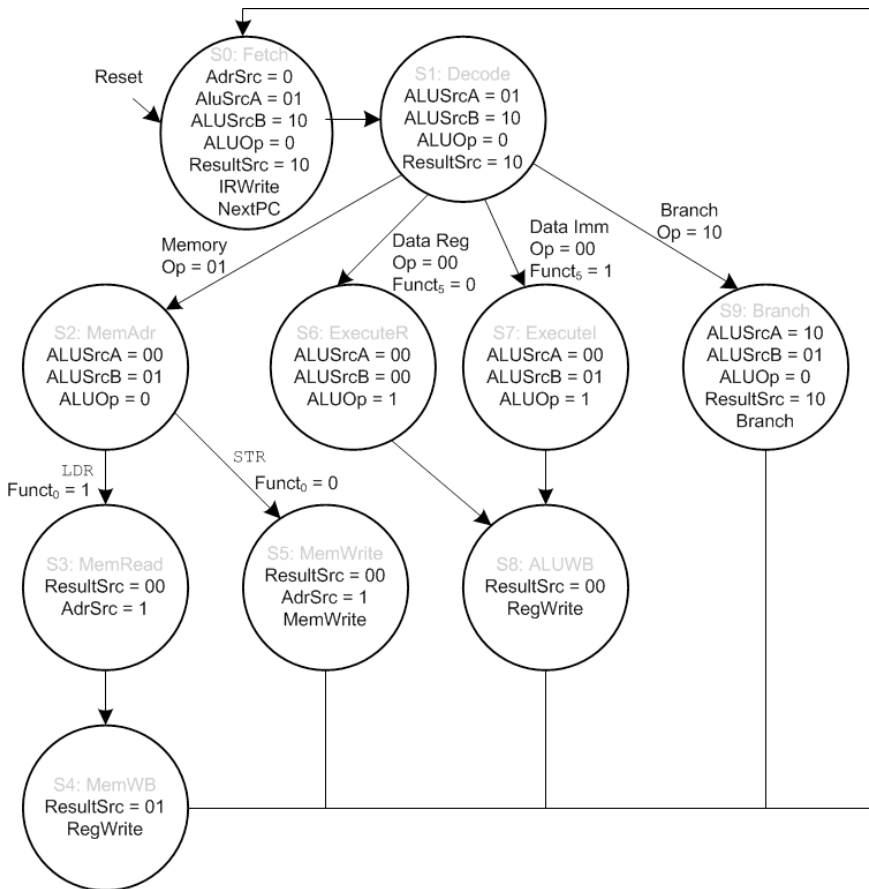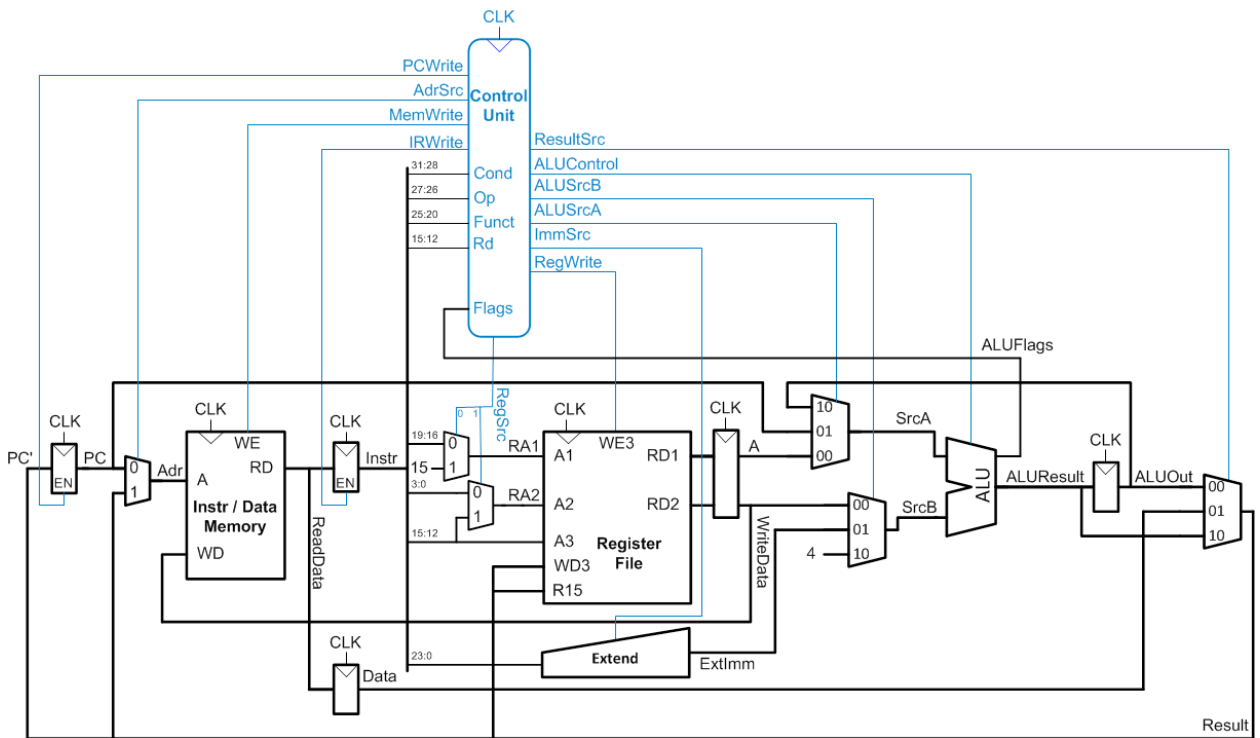J. Spjut

Problem Set 9

1) Problems

1. Modify the multi-cycle ARM processor to add the functionality required to implement $R_{rshiftr}$-Type instructions. Label all added signals and describe the circuits in each block added to the data-path. Also show any changes required for the controller FSM.

2. Find the new cycle-time for a multi-cycle and calculate the execution time for 100 billion instructions for the ARM processor based on the timings in Table 0.5 in chapter 7.3 if a new ALU design reduces the ALU delay by 20ps. Use the same instruction mix as example 7.5 in the text.

3. How many cycles are required to run the following program on the multi-cycle ARM processor? What is the CPI of this program?

```
        MOV R0, #5
while:
        CMP R0, #0
        BEQ done
        SUB R0, R0, #1
        B while
done:
```

2) Time

Please indicate how many hours you spent on this problem set. This will not affect your grade (unless entirely omitted), but will be helpful for calibrating the workload for next semester's class.

CLK

PCWrite
AdrSrc
MemWrite
IRWrite

Control Unit

Cond   31:28
Op   27:26
Funct   25:20
Rd   15:12

Flags

ResultSrc
ALUControl
ALUSrcB
ALUSrcA
ImmSrc
RegWrite

ALUFlags

CLK PC' PC Adr WE RD A **Instr / Data Memory** WD ReadData

CLK Instr EN

RegSrc

19:16 RA1 A1 WE3 RD1
15 RA2 A2 RD2
3:0
15:12 A3 **Register File** WD3 R15

WriteData

CLK A SrcA 10 01 00

ALU ALUResult CLK ALUOut 00 01 10

SrcB 00 01 10 4

CLK Data 23:0 **Extend** ExtImm

Result

State	Datapath µOp
Fetch	Instr ←Mem[PC]; PC ← PC+4
Decode	ALUOut ← PC+4
MemAdr	ALUOut ← Rn + Imm
MemRead	Data ← Mem[ALUOut]
MemWB	Rd ← Data
MemWrite	Mem[ALUOut] ← Rd
ExecuteR	ALUOut ← Rn op Rm
ExecuteI	ALUOut ← Rn op Imm
ALUWB	Rd ← ALUOut
Branch	PC ← ALUOut + offset

Reset

S0: Fetch
AdrSrc = 0
AluSrcA = 01
ALUSrcB = 10
ALUOp = 0
ResultSrc = 10
IRWrite
NextPC

S1: Decode
ALUSrcA = 01
ALUSrcB = 10
ALUOp = 0
ResultSrc = 10

Memory
Op = 01

Data Reg
Op = 00
$Funct_5 = 0$

Data Imm
Op = 00
$Funct_5 = 1$

Branch
Op = 10

S2: MemAdr
ALUSrcA = 00
ALUSrcB = 01
ALUOp = 0

S6: ExecuteR
ALUSrcA = 00
ALUSrcB = 00
ALUOp = 1

S7: ExecuteI
ALUSrcA = 00
ALUSrcB = 01
ALUOp = 1

S9: Branch
ALUSrcA = 10
ALUSrcB = 01
ALUOp = 0
ResultSrc = 10
Branch

LDR
$Funct_0 = 1$

STR
$Funct_0 = 0$

S3: MemRead
ResultSrc = 00
AdrSrc = 1

S5: MemWrite
ResultSrc = 00
AdrSrc = 1
MemWrite

S8: ALUWB
ResultSrc = 00
RegWrite

S4: MemWB
ResultSrc = 01
RegWrite