Project Presentation

Spartak Gevorgyan, Vladislav Semenyutin, Shohin Abdulhamidov

Assembly Program

```
MOV R4, #0      ; register to store the sum
MOV R0, #0      ; index i, will be used in the loop
MOV R3, #16     ; final size of the array, 4 words
LDR R1, =A      ; load array A into register 1
LDR R2, =B      ; load array B into register 2
LDR R7, =C      ; load array C into register 7, it will be the empty array where results will be saved
LOOP:
    LDR R5, [R1], #4     ; load value of array A into register 5 and update base register by R1 = R1 + 4
    LDR R6, [R2], #4     ; load value of array A into register 6 and update base register by R2 = R2 + 4
    ADD R4, R5, R6       ; add values from register 5 and 6 and place the result in register 4
    STR R4, [R7], #4     ; store value into R7 and update base register by R7 = R7 + 4
    ADD R0, #4           ; increase counter by 1 word = 4 bytes
    CMP R0, R3           ; check if we are done
    BLT LOOP             ; if not done loop again
```

Instructions

Single Data Transfer

Assembly Language Symbol	Instruction
LDR	Load register from memory
STR	Store register to memory

Branch

Assembly Language Symbol	Instruction
BLT	Branch if less than

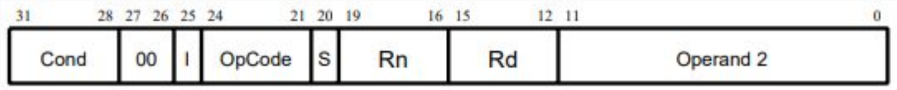
Data Processing

Assembly Language Symbol	Instruction	OpCode
MOV	Move register or constant	1101
ADD	Addition of operands from registers	0000
CMP	Compare values in register	1010
SUB	Subtraction of operands from registers	0010
ORR	Bitwise OR operation	1100

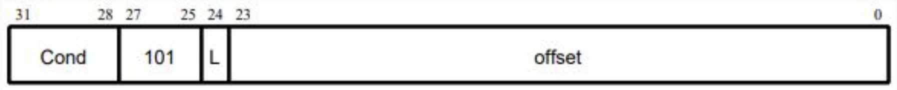
Instructions

In general we could divide all our instructions into 3 categories. We had to design microarchitecture which could work for all of these 3 categories of instructions.

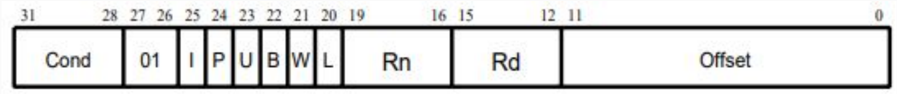
Data Processing



Branch



Load/Store



Data Processing, Branch, Load/Store

Data Processing:

- We first load instruction from PC into instruction memory and increment the PC by 4 bytes.
- We read values out of registers RS and RT and ALU operates on data from registers using the Bits 5-0 from instructions which are the operation opcodes.
- The ALU result then gets written into the Write Register.

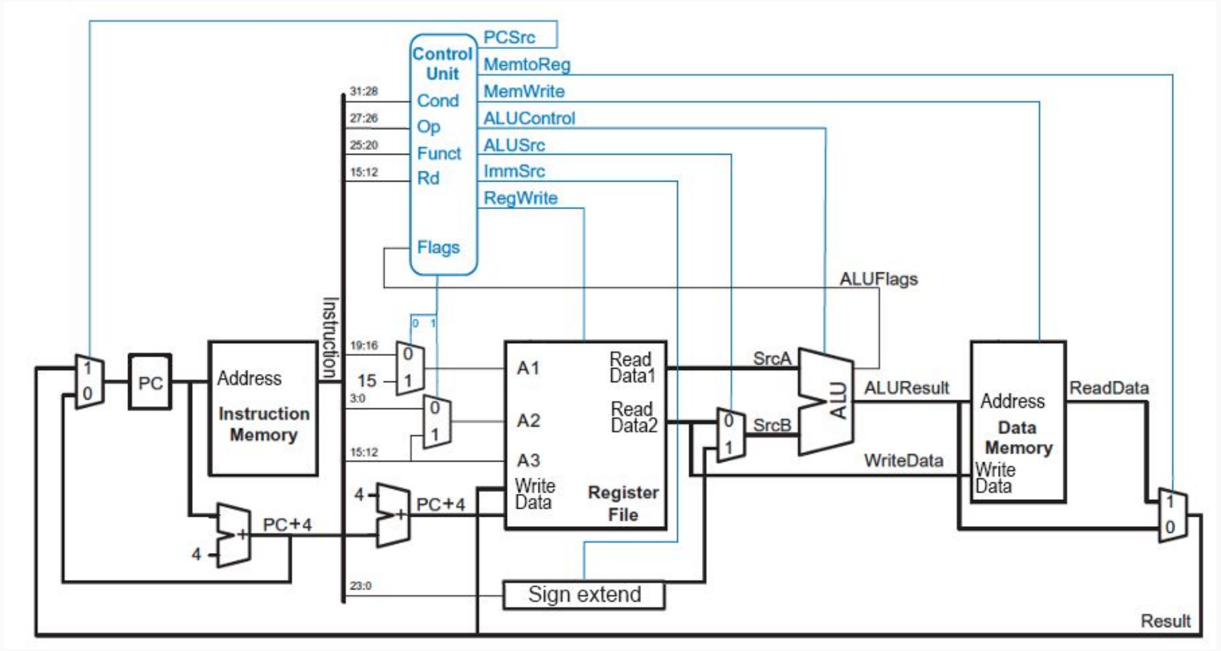
Branch:

- We load instruction from PC into instruction memory.
- We read values out of registers RS and RT.
- ALU then subtracts those two values to see if it is zero. In case of zero we will send one out of zero flag and because we are also branching, our ‘and’ gate will be two 1’s and it outputs one.
- The address of the next instruction will be the adder sums PC + 4 plus immediate value shifted left by two bits.

Load/Store:

- We first load instruction from PC into instruction memory and increment the PC by 4 bytes.
- We read base address value from register RS and read value out of RT register into write register.
- ALU adds the base address from register RS to the sign-extended lower 16 bits of the instruction which is the offset.
- At the end we retrieve or write data from the memory unit into the register file, where the register index in the RT.

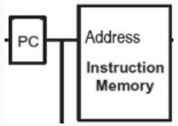
ARM Datapath

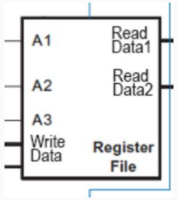


Data Processing Test

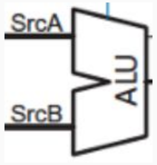
Steps:

1. We load instruction from PC into Instruction Memory.

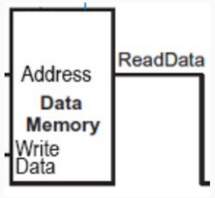


2. Let's consider we have the instruction ADD R4, R4, R5 where the binary will be 1110 00 0 0100 0 0010 0010 0000 00000011
3. 

We load address [19-16] into A1 and address [15-12] into A3. We load address [3-0] into A2.

4. 

Our control unit will send signal to ALU with OpCode binary [24-21] which is 0100 = ADD. Thus only we will have ALUControl on, everything else fo control unit will be set off. We load address [19-16] into A1 and address [15-12] into A3. We load address [3-0] into A2.

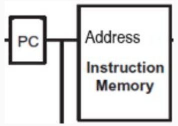
5. 

At the end ALU result will be written into R4 and the new instruction will be loaded.

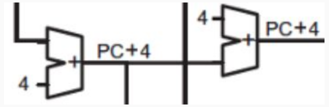
Branch Test

Steps:

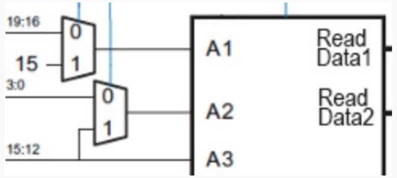
1. We load instruction from PC into Instruction Memory.



2. PC + 8 is read from the first port of the register file.



3. Therefore, a multiplexer is needed to choose R15 as the RA1 input.

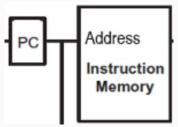


4. This multiplexer is controlled by another bit of RegSrc, choosing Instr [19:16] for most instructions but 15 for B.
5. MemtoReg is set to 0 and PCSrc is set to 1 to select the new PC from ALUResult for the brunch.
6. Thus, we do branch instruction and load the next instruction into memory.

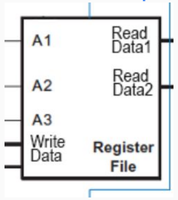
Load/Store Test

Steps:

1. The first step is to read this instruction from instruction memory.



2. The instruction memory reads out, or fetches, the 32-bit instructions, labeled Instruction.
3. The next step is to read the source register containing the base address.



4. This bits of the instruction are connected to the address input of one of the register file ports, A1.
5. The register file reads the register value onto RD1. The LDR instruction also required on offset.
6. The offset is stored in the immediate field of the instruction [11-0]
7. It is an unsigned value, so it must be zero-extended to 32 bits.
8. The 32-bit value is called Sign Extend.
9. Zero extension simply means preparing leading zeros:
    a. [31-12] = 0 and [11-0] = [11-0]

Truth Table

Signal name	**R-format**	**lw**	**sw**	**beq**
RegDst	1	0	X	X
ALUSrc	0	1	1	0
MemtoReg	0	1	X	X
RegWrite	1	1	0	0
MemRead	0	1	0	0
MemWrite	0	0	1	0
Branch	0	0	0	1
ALUOp1	0	0	0	1
ALUOp0	0	0	0	1