Advanced Computer Architecture

Lecture 19

Reading Material

Vincent P. Heuring&Harry F. Jordan — Chapter 5
Computer Systems Design and Architecture — 5.1.3

Summary

- Pipelined Version of the SRC
- Adapting SRC instructions for Pipelined Execution
- Control Signals for Pipelined SRC

Pipelined Version of the SRC

In this lecture, a pipelined version of the SRC is presented. The SRC uses a five-stage pipeline. Those five stages are given below:

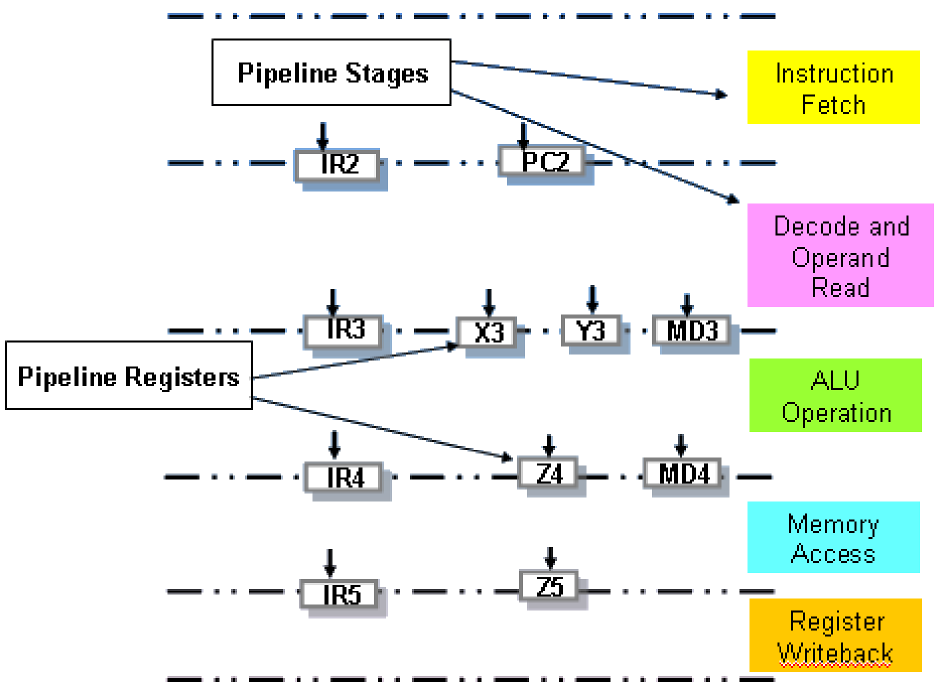
1. Instruction Fetch
2. Instruction decode/operand fetch
3. ALU operation
4. Memory access
5. Register write

As shown in the next diagram, there are several registers between each stage.

After the instruction has been fetched, it is stored in **IR2** and the incremented value of the program counter is held in **PC2**. When the register values have been read, the first register value is stored in **X3**, and the second register value is stored in **Y3**. **IR3** holds the opcode and ra. If it is a store to memory instruction, **MD3** holds the register value to be stored.

After the instruction has been executed in the ALU, the register **Z4** holds the result. The op-code and ra are passed on to **IR4**. During the write back stage, the register **Z5** holds the value to be stored back into the register, while the op-code and ra are passed into **IR5**. There are also two separate memories and several multiplexers involved in the pipeline operation. These will be shown at appropriate places in later figures.

The number after a particular register name indicates the stage where the value of this register is used.



Adapting SRC Instructions for Pipelined Execution

As mentioned earlier, the SRC instructions fall into the following three categories:

1. ALU Instructions
2. Load/Store instructions
3. Branch Instructions

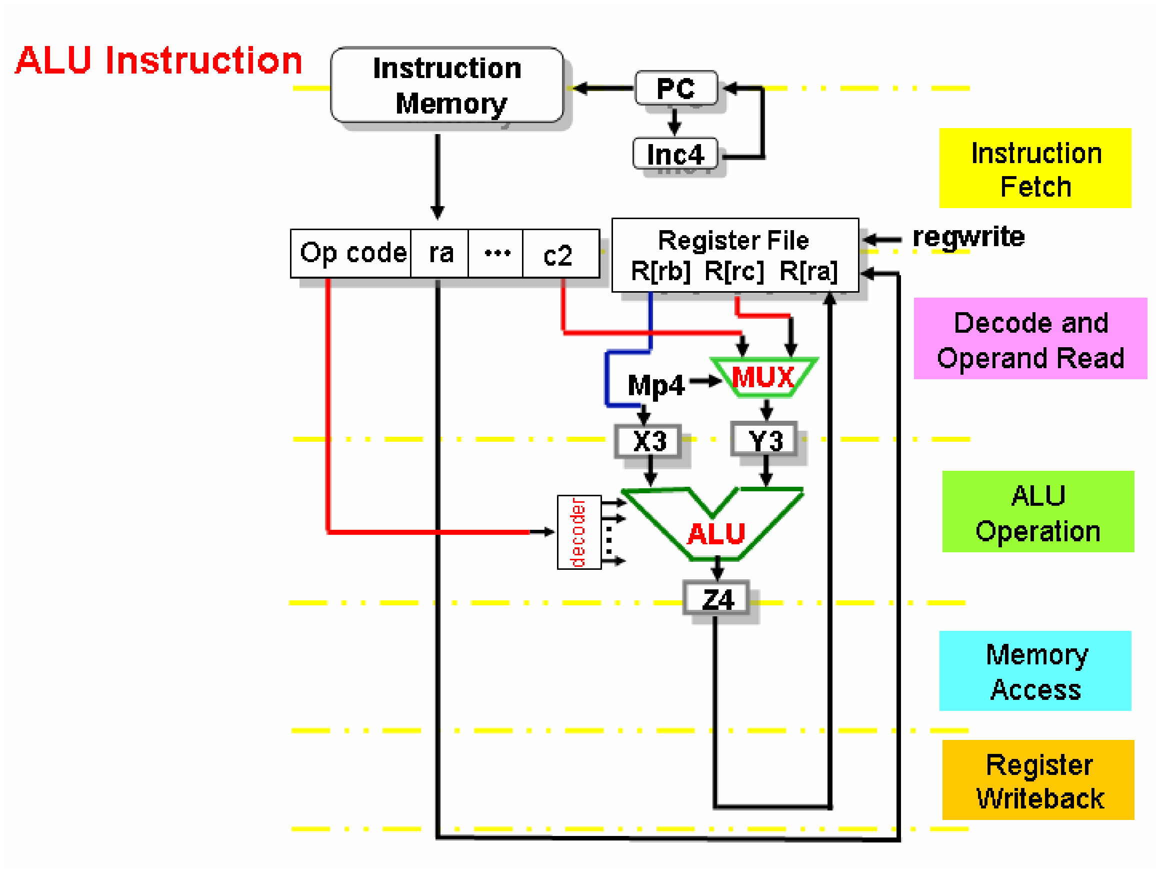
We will now discuss how to design a common pipeline for all three categories of instructions.

1. ALU instructions

ALU instructions are usually of the form:

op-code ra, rb, rc
or
op-code ra, rb, constant.

In the diagram shown, X3 and Y3 are temporary registers to hold the values between pipeline stages. X3 is loaded with operand value from the register file. Y3 is loaded with either a register value from the register file or a constant from the instruction. The operands are then available to the ALU. The ALU function is determined by decoding the op-code bits. The result of the ALU operation is stored in register Z4, and then stored in the destination register in the register write back stage. There is no activity in the memory access stage for ALU instructions. Note that Z5, IR3, IR4, and IR5 are not shown explicitly in the figure. The purpose of not including these registers is to keep the drawing simple. However, these registers will transfer values as instructions progress through the pipeline. This comment also applies to some other figures in this discussion.

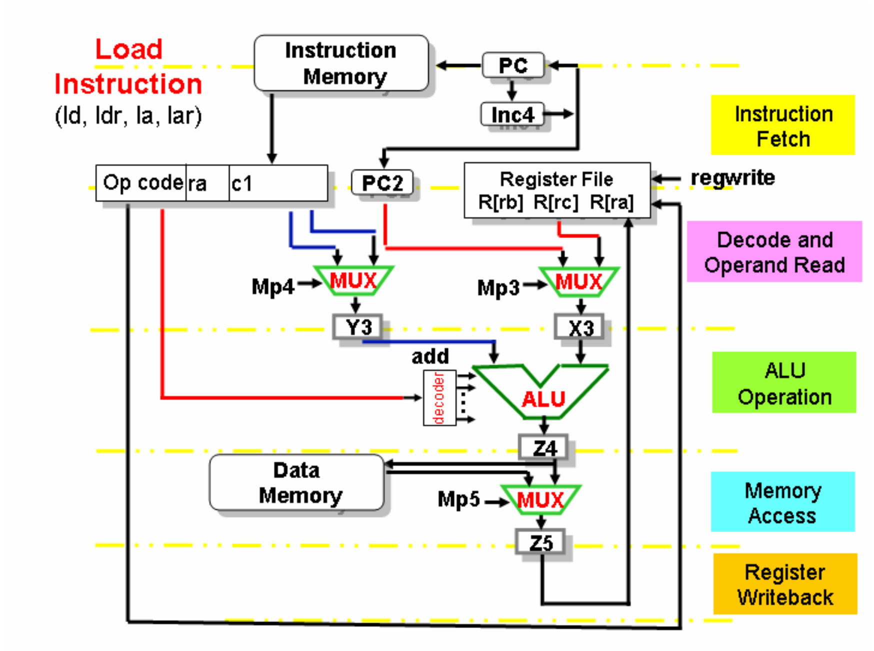


2. Load/Store instructions

Load/Store instructions are usually of the form:

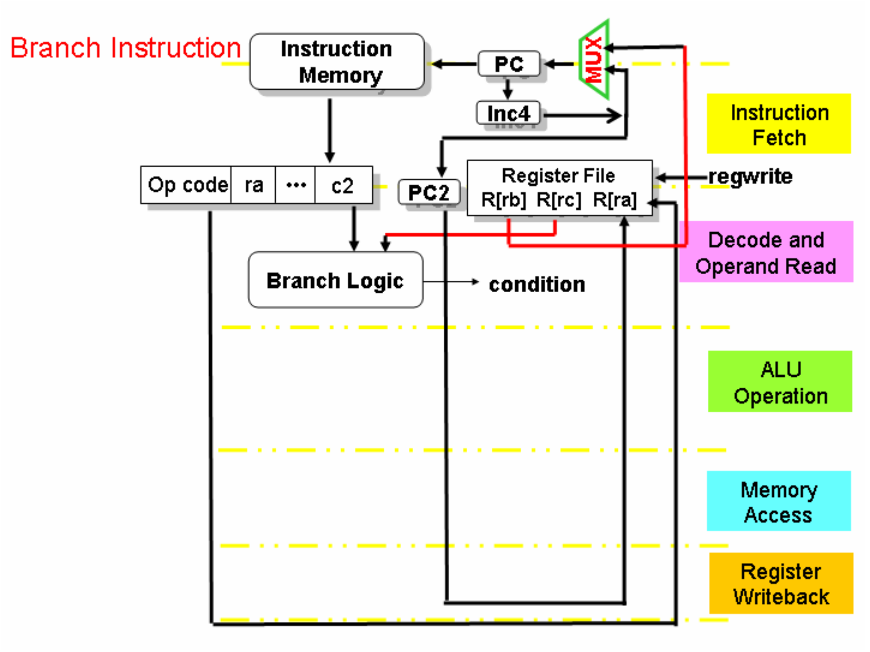
op-code ra, constant(rb)

The instruction is loaded into **IR2** and the incremented value of the PC is loaded in **PC2**. In the next stage, **X3** is loaded with the value in **PC2** if the relative addressing mode is used, or the value in **rb** if the displacement addressing mode is used. Similarly, **C1** is transferred to **Y3** for the relative addressing mode, and **c2** is transferred to **Y3** for the displacement addressing mode. The store instruction is completed once memory access has been made and the memory location has been written to. The load instruction is completed once the loaded value is transferred back to the register file. The following figure shows the schematic for a load instruction. A similar schematic can be drawn for the store instruction.



3. Branch Instructions

Branch Instructions usually involve calculating the target address and evaluating a condition. The condition is evaluated based on the c2 field of the IR and by using the value in R[rc]. If the condition is true, the PC is loaded with the value in R[rb], otherwise it is incremented by 4 as usual. The following figure shows these details.

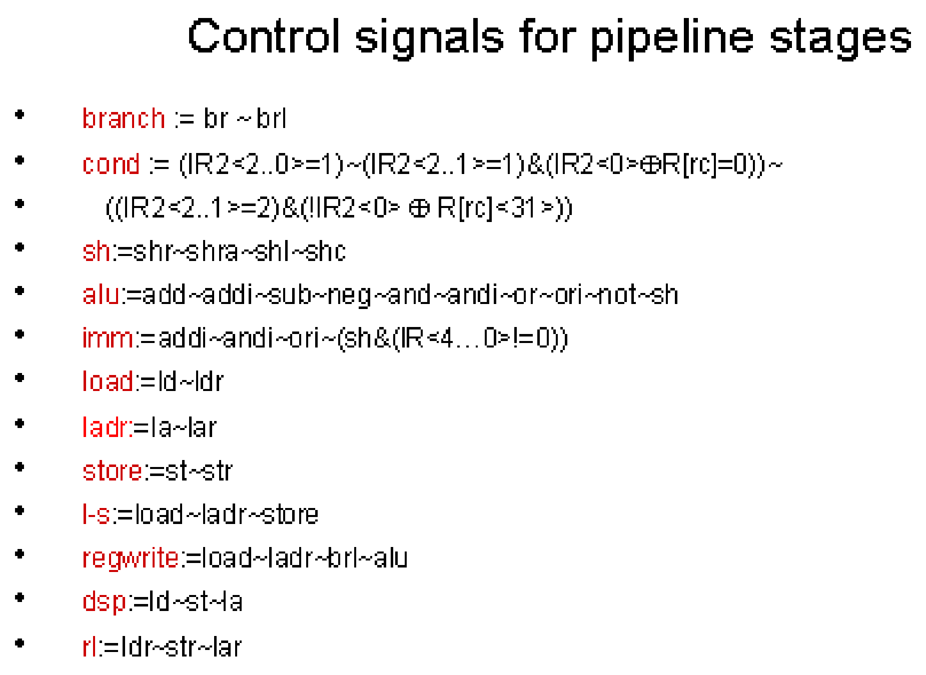


The complete pipelined data path

The pipelined data path implementation diagrams shown earlier for the three SRC instruction categories must be combined and refined to get a working system. These details get complicated very quickly. A detailed combined diagram is shown in Figure 5.7 of the text book.

Control Signals for the Pipelined SRC

We define the following signals for the SRC by grouping similar op-codes:



Control signals for pipeline stages

- branch := br ~ brl
- cond := (IR2<2..0>=1)~(IR2<2..1>=1)&(IR2<0>⊕R[rc]=0))~
- ((IR2<2..1>=2)&(!IR2<0> ⊕ R[rc]<31>))
- sh:=shr~shra~shl~shc
- alu:=add~addi~sub~neg~and~andi~or~ori~not~sh
- imm:=addi~andi~ori~(sh&(IR<4…0>!=0))
- load:=ld~ldr
- ladr:=la~lar
- store:=st~str
- l-s:=load~ladr~store
- regwrite:=load~ladr~brl~alu
- dsp:=ld~st~la
- rl:=ldr~str~lar

In most cases, the signals defined above are used in the same stage where they are generated. If that is not the case, a number used after the signal name indicates the stage where the signal is generated.

Using these definitions, we can develop RTL statements for describing the pipeline activity as well as the equations for the multiplexer select signals for different stages of the pipeline. This is shown in the next diagram.

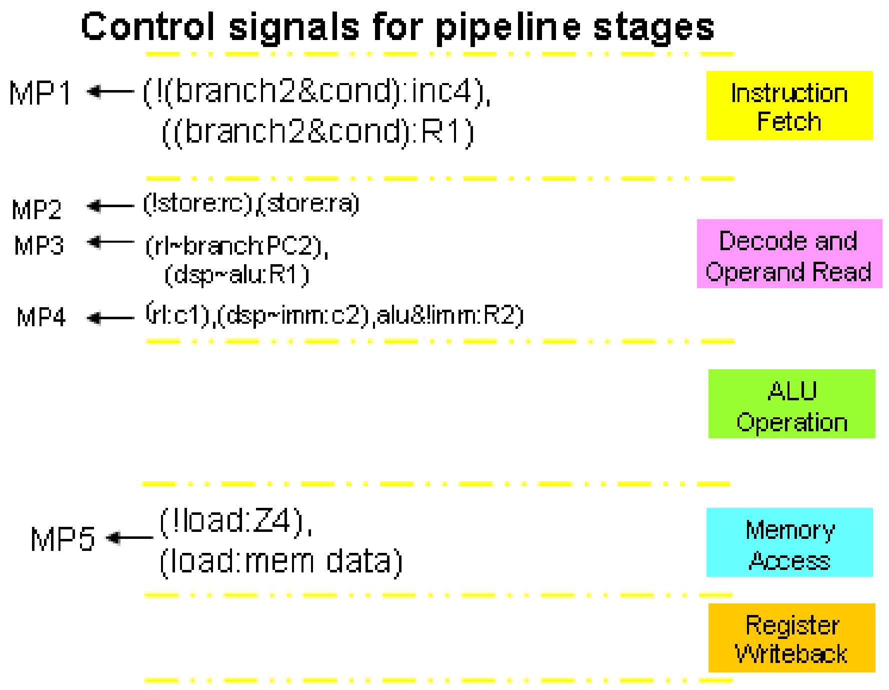
Control Signals for different pipeline Stages

Consider the RTL description of the Mp1 signal, which controls the input to the PC. It simply means that if the branch and cond signals are not activated, then the PC is incremented by 4, otherwise if both are activated then the value of R1 is copied in to the PC.

The multiplexer Mp2 is used to decide which registers are read from the register file. If the store signal is activated then R[rb] from the instruction bits is read from the register file so that its value may be stored into memory, otherwise R[rc] is read from the register file.

The multiplexer Mp3 is used to decide which registers are read from the register file for operand 2. If either rl or branch is activated then the updated value of PC2 is transferred to X3, otherwise if dsp or alu is activated, the value of R[ra] from the register file is transferred to the x3. In the same way, multiplexer Mp4 is used to select an input from Y3.

In the same way, multiplexer Mp4 is used to select an input for Y3.


Control signals for pipeline stages
MP1 ← (!(branch2&cond):inc4), ((branch2&cond):R1)
Instruction Fetch
MP2 ← (!store:rc),(store:ra)
MP3 ← (rl~branch:PC2), (dsp~alu:R1)
MP4 ← (rl:c1),(dsp~imm:c2),alu&!imm:R2)
Decode and Operand Read
ALU Operation
MP5 ← (!load:Z4), (load:mem data)
Memory Access
Register Writeback

The multiplexer MP5 is used to decide which value is transferred to be written back to the register file. If the load signal is activated data from memory is transferred to Z5, however if the load signal is not activated then data from Z4 (which is the result of ALU) is transferred to Z5 which is then written back to the register file.