CS147 - Lecture 14

Kaushik Patra
(kaushik.patra@sjsu.edu)

1

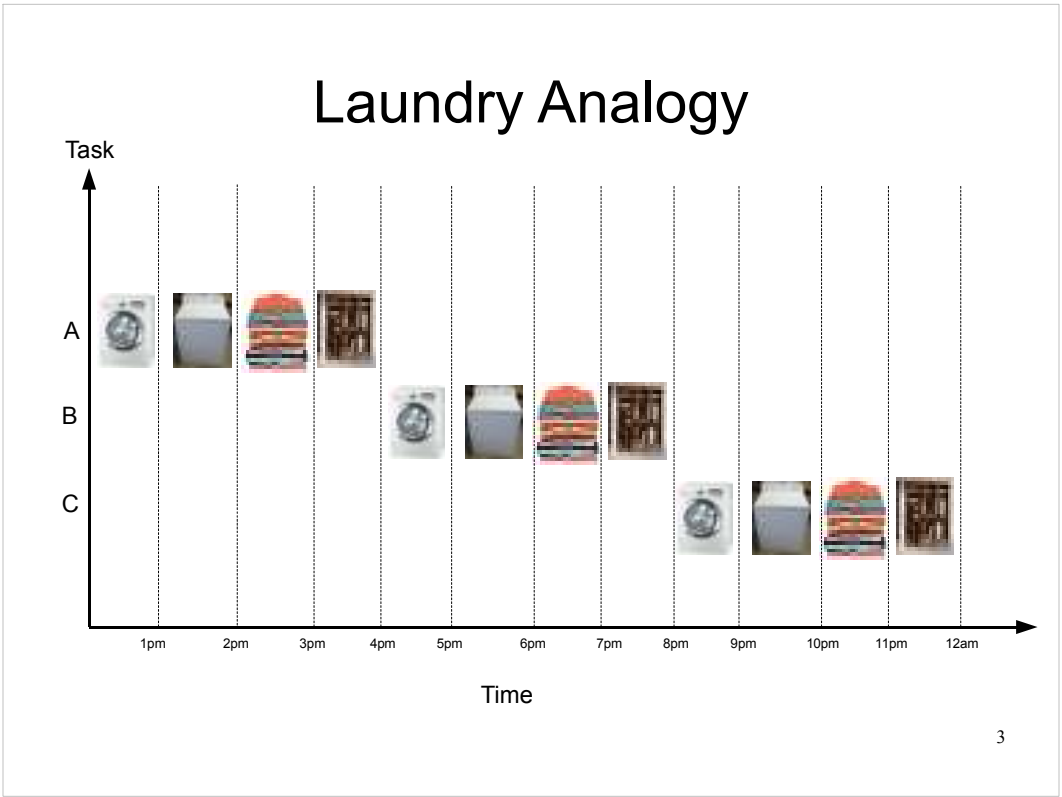
- Performance Improvement using Pipelining
- Instruction Set Architecture for Pipeline
- Pipeline Hazards

Reference Books:

1) *Chapter 6 of 'Computer Organization & Design' by Patterson & Hennessy*

Pipelining for Performance ...

2



- The entire laundry task can be broken down into four steps – washing, drying, cloth folding and storing. Let's say each step takes 1 hour each. Hence for three person Andy A, Bobby B and Chandra C the entire laundry process takes four hours each.

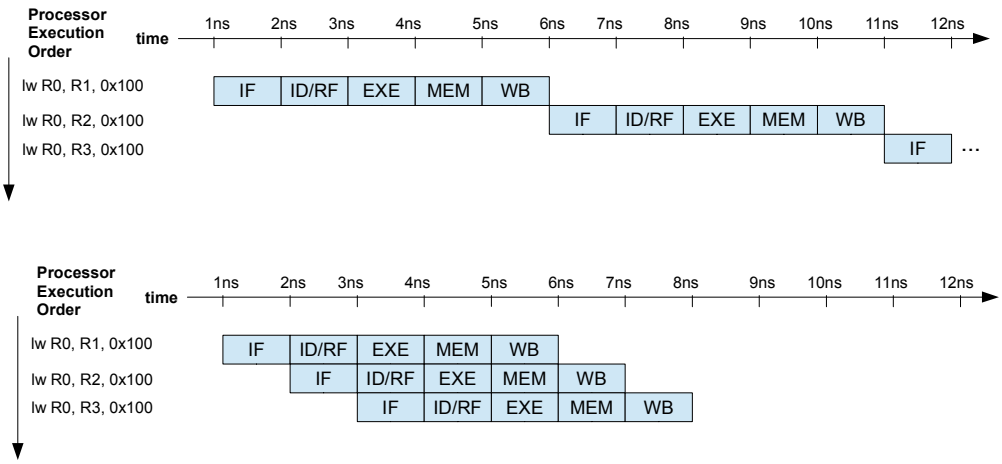
- If A, B and C does their laundry task serially one after another, they will take total 4*3=12 hours to finish the entire tasks for all of them.

- However, once A is done with the washer, it remains available for entire time in which A completes rest of the task. Hence someone else can start using the washer in parallel with A. Similar for drying, folding and storing sub-tasks.

- Therefore A, B and C do not need to wait till other is completely done with their laundry tasks. As soon as any resource is available for their sub-task, they can start it. In that way the total amount of time needed to finish laundry tasks for all 3 people. This is speeding up the process by increasing throughput without decreasing time of individual step.

- This is the basic idea of pipelining of tasks – where sub-tasks can be started as soon as resource is available.

Processor Pipeline



5

- MIPS and similar designs (like one we are doing to implement a processor to support cs147sec05 ISA) go through five stages to execute an instruction. These five stages are instruction fetch (IF), instruction decode and register fetch (ID/RF), execute (EXE), memory access (MEM), and write back (WB).

- All these five stages uses independent resources.
  - IF : Instruction memory to read
  - ID/RF : Register File to read
  - EXE : ALU for the computation of result / memory address
  - MEM : Data memory to read or write.
  - WB : Register file to write.

- Since stages uses independent resources, instruction execution can be pipelined similar to the pipelined laundry process. As soon as instruction fetch is done for one instruction, IF for next instruction can be done. This apples to other stages too. As a result we are reducing effective completion time of each instruction without reducing actual execution time of the instruction.

Performance Improvement

- For a balanced n-stage pipeline structure the maximum speedup obtained is n.

$$Speedup_{max} = \frac{E_{non-pipelined}}{E_{pipelined}} = n$$

- For a balanced n-stage pipeline CPI is improved by n times.
- Pipeline increases performance by *increasing instruction throughput* as oppose to decreasing the execution time of individual instruction.

6

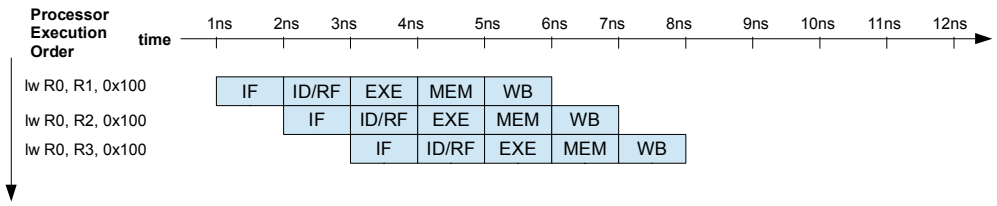
- A balanced pipeline structure is an implementation where each stage takes same amount of time.
- For a n-stage balanced pipeline, maximum speed up obtained is n. However, often it is not the case due to data dependencies (we call it hazards in terms of pipeline structure) between instruction where one instruction execution must be held till one of the previous instruction is completed because current instruction uses result form previous instruction.
- In another word, we can say that this n-time improvement in execution time is coming from improvement of CPI metric by n in n-stage pipeline. Only the first instruction will take n cycles to be completed (CPI = n for the instruction executed). Every other successive instructions will be effectively done in 1 clock cycle, assuming there is no data conflict / dependencies, because (n-1) stages are already done in parallel with previous instruction. Hence for a large program, with million of instruction the effective CPI tends to be 1. Hence the CPI metric is improved by n with n-stage pipeline.
- The other observation is that performance of the processor is improved by increasing the throughput of instructions executed. Individual execution time is unchanged in this architecture.

Performance Improvement

- However, in the example of slide 5, the speedup is ~2x.

$$E_{non-pipelined} = 15\text{ns},\ E_{pipelined} = 7\text{ns}$$

$$\Rightarrow Speedup = \frac{15}{7} \approx 2$$



7

- If we calculate speed in the example in slide 5, we'll see that the improvement is about 2x. However, with a 5 stage pipeline architecture, it should be about 5x. This is because we are analyzing a small amount of instructions. Pipeline speedup becomes closer to 5, in this case, as we execute more number of instructions.

Performance Improvement

- Pipeline performance improvement is for large programs. Let's redo the calculation with 1K instruction.

$$E_{non-pipelined} = 5000\text{ns}$$

$$E_{pipelined} = ((1*5)+(1000-1)*1)\,ns = 5+999 = 1004$$

$$\Rightarrow Speedup = \frac{5000}{1004} \approx 4.98 \rightarrow 5$$

8

- Let's assume this 5-stage pipelined design executes 1K instruction. For a non-pipelined design, since CPI is 5, to execute 1K instructions it will take 5K cycles. Assuming each cycle takes 1ns, execution time is 5000ns. For the pipelined design, the first instruction will take 5 cycle and rest of the 999 instructions will take 1 cycle to complete. Hence total cycle needed is (5+999) = 1004. Therefore the speedup is about 4.98. As we increase the number of instructions the performance gain will tend to be 5x.

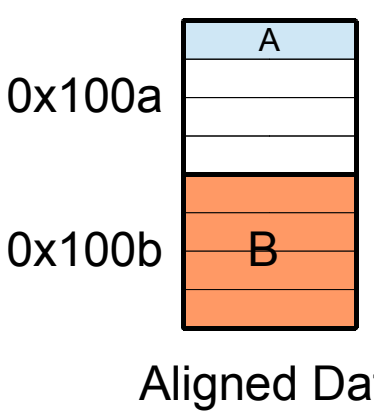
Instruction Set Architecture for Pipeline ...

ISA for Pipeline

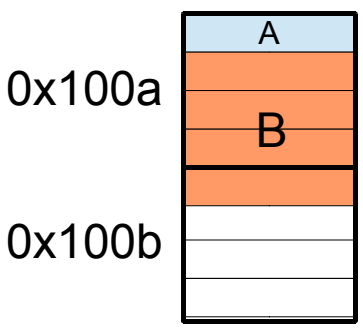
- Pipelining is a complex idea to be implemented on hardware. ISA should be kept simple enough to have overly complex control circuit.
  - Equal length Instructions
  - Supports fewer instruction format
  - Limited instructions to access data memory
  - Operands must be aligned in memory

10

- Equal length of instruction makes it easy to fetch the instruction from memory. Unlike to any CISC style ISA, where number of bytes to be fetched to completely read an instruction depends on the opcode of the instruction fetched, RISC style ISA always enforce equal length of instruction so one simple reading of a word or double word from memory is sufficient to complete reading of an instruction.
- Supporting fewer instruction type will reduce complexity of the instruction decoder and the data path for the instruction.
- Limited instruction to access data memory will reduce data path complexity.
- RISC style ISA assumes data is aligned within memory. This means single data access of any type (byte, word, double word, quad word) will need single memory access. For a double word accessible memory (32-bit data) there can be data storage for a byte and a double word as followings. For an unaligned memory, there needs to be two memory read to retrieve data B.



Aligned Data



Unaligned Data

Pipeline Hazards ...

11

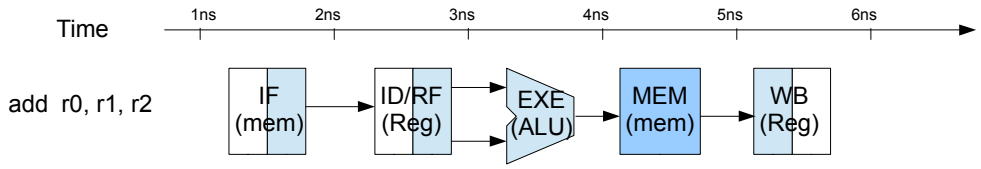
Types of Hazards

- Structural Hazard
- Data Hazard
- Control Hazard

12

- Structural hazards occurs when two different stages (of different instruction in pipeline) wants to access same resource (memory or register file or ALU).
- Data hazard occurs when a later instruction depends on the result of a previous instruction, which is not completed yet.
- Control hazard occurs when next instruction address depends on the previous instruction result or a decision in the current instruction (e.g. beq r[1], r[2], 0x01a2 – the next instruction address depends on whether r[1] and r[2] are equal or not).

Pipeline Resource Identification



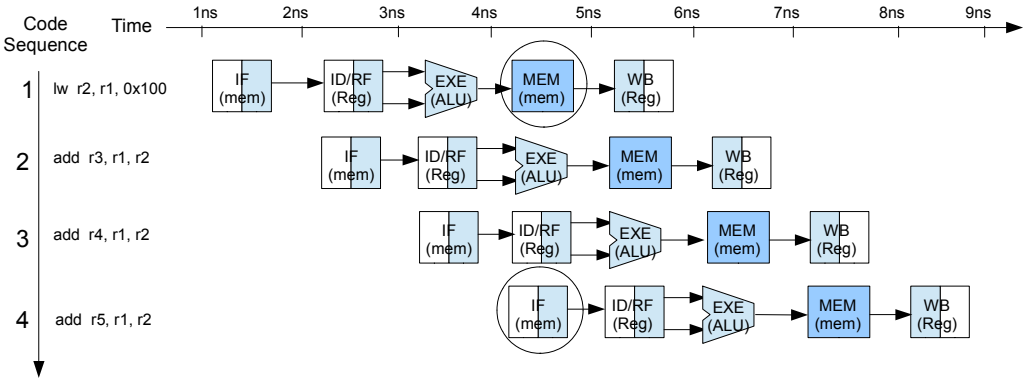
- IF → Memory Read
- ID/RF → Register File Read
- EXE → ALU operation
- MEM → Memory Read/Write
- WB → Register File Write

13

- As discussed earlier the resources needed for each stages as as following.
  - IF : Instruction memory to read
  - ID/RF : Register File to read
  - EXE : ALU for the computation of result / memory address
  - MEM : Data memory to read or write.
  - WB : Register file to write.

- In the pipeline resource diagram, memory is shown with half filled at right hand side in IF stage. This mean the memory or storage is accessed for reading. Similarly at ID/RF stage the register file is accessed for reading. This is marked with half filled register file block at right hand side. At the execution stage ALU is used as a whole. At the MEM stage the memory can either be written or read (depending on lw or sw instruction), hence the whole 'mem' box is filled up. At the WB stage the result is written back to register file, hence the left hand side of the register file box is filled up.

Structural Hazard - 1



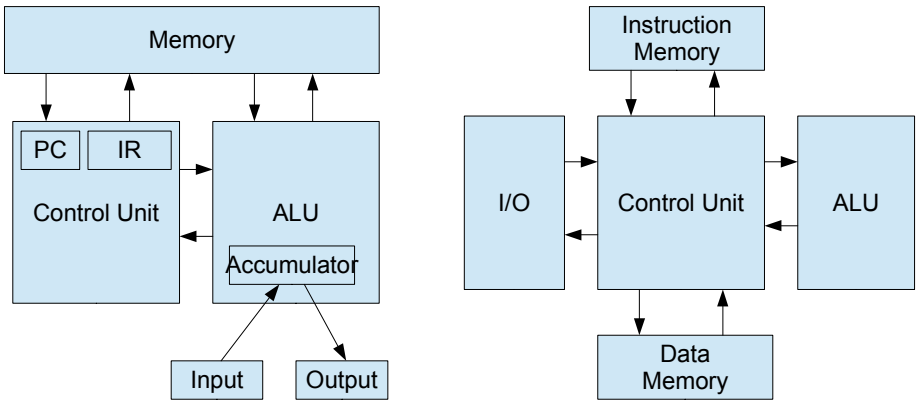
- @ 4ns Instruction-1 is accessing memory for reading data.
- @ 4ns Instruction-4 is accessing memory for reading instruction.

14

- In the given instruction sequence, MEM stage of instruction 1 is at conflict with IF of instruction 4, because both of this stage is accessing same memory block at different address. Since the memory, we are using in the implementation, supports single operation at a time only, these two stages (MEM@1 and IF@4) are in conflict.

Resolving Structural Hazard - 1

- Von Neumann architecture can not support pipelining.



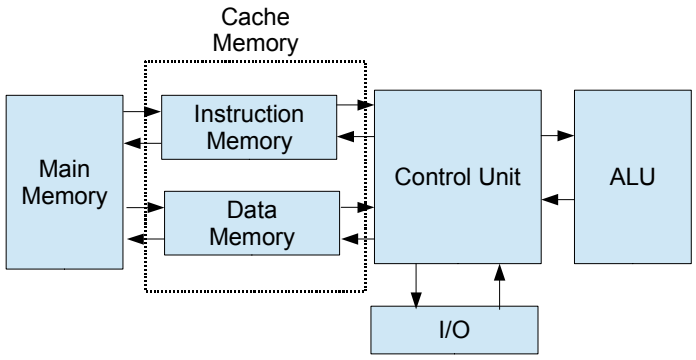
- Harvard architecture can support pipelining.

15

- To resolve such structural hazard as in slide 14, we must differentiate between instruction memory and data memory. Remember that the MEM@1 is accessing the data part of the memory and IF@4 is accessing the instruction part of the memory (in slide 14). Hence separating memory and data into two different independent memory block will help to resolve this structural hazard.

- Having separate instruction and data memory is the essence of the Harvard architecture. We can also safely claim that the Von Neumann architecture can not support the pipeline architecture since instruction and data memory is same in this architecture.

Resolving Structural Hazard - 1



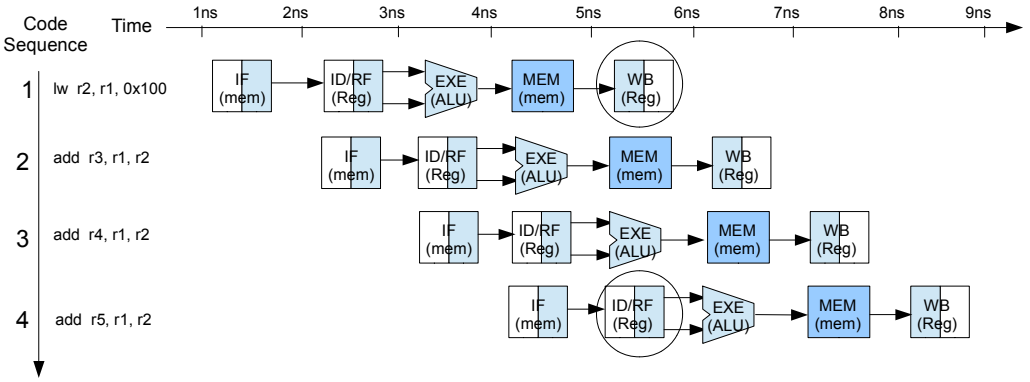
- Introduction of Cache memory in memory hierarchy mixed Von Neumann and Harvard architecture.

16

- In practice, hybrid approach of Harvard and Von Neumann architecture is taken to resolve this structural hazard as in slide 14. A main memory is still maintained to store both data and instructions. However, main memory is connected to some intermediate memory structures separating instruction and data. Relevant parts of the data and instructions are copied from main memory into these memories and results are copied back into the main memory from the data memory (if there is any data written onto the separate data memory).

- These intermediate memories (separating data and instructions) are known as cache in memory hierarchy structure. We'll do some detailed discussion on this cache mechanism when we learn about memory hierarchy.

Structural Hazard - 2

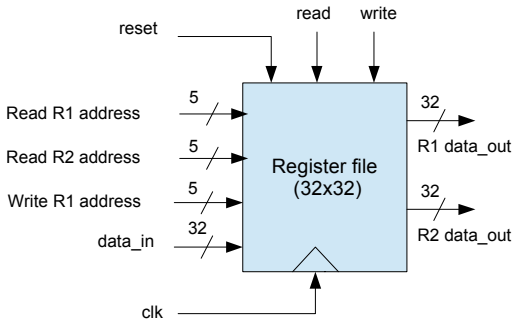


- @ 5ns Instruction-1 is accessing register file for writing.
- @ 5ns Instruction-4 is accessing register file for reading.

17

- In the given instruction sequence, WB stage of instruction 1 is at conflict with ID/RF of instruction 4, because both of this stage is accessing same register file for different operation. Since the register file, we are using in the implementation, supports single operation at a time only, these two stages (WB@1 and ID/RF@4) are in conflict.

Resolving Structural Hazard - 2



Old Operation Table

read	write	operation
0	0	Hold
0	1	Write
1	0	Read
1	1	Hold

New Operation Table

read	write	operation
0	0	Hold
0	1	Write
1	0	Read
1	1	Read & Write

- To resolve this type of structural hazard we must support two operation (read and write) at one time.

18

- So far, in our project and discussion, we were using a register file which support one operation at a time. If both read and write are 00 or 11, it will hold the previously read data onto the output ports of the register file.

- To resolve the structural hazard as presented in slide 17, we must enhance the register file so that if both read and write is 1, it will perform both read and write operation. Since the read and write addresses are separated and the data-in / data-out are also separate, it is easier to implement such operation where both read and write can be done in one single operation.

CS147 - Lecture 14

Kaushik Patra
(kaushik.patra@sjsu.edu)

19

- Performance Improvement using Pipelining
- Instruction Set Architecture for Pipeline
- Pipeline Hazards

Reference Books:

1) *Chapter 6 of 'Computer Organization & Design' by Patterson & Hennessy*