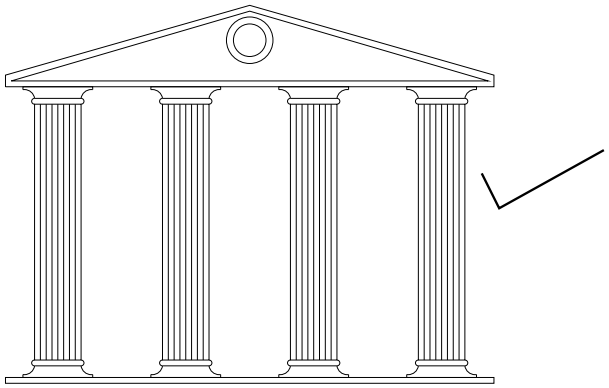
Review of Computer Architecture

In this course we are concerned with systems architecture:



- The arrangement of computer components
- The mechanisms for connecting these components together
- How the problems change as we have multiple copies of certain components

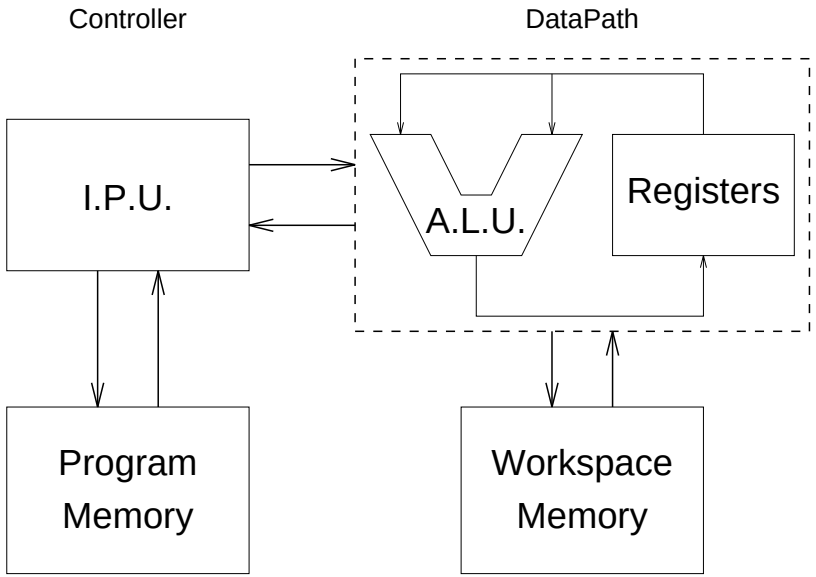
We are less concerned with the components themselves:

- We shall not be discussing the design of a better multiplier.

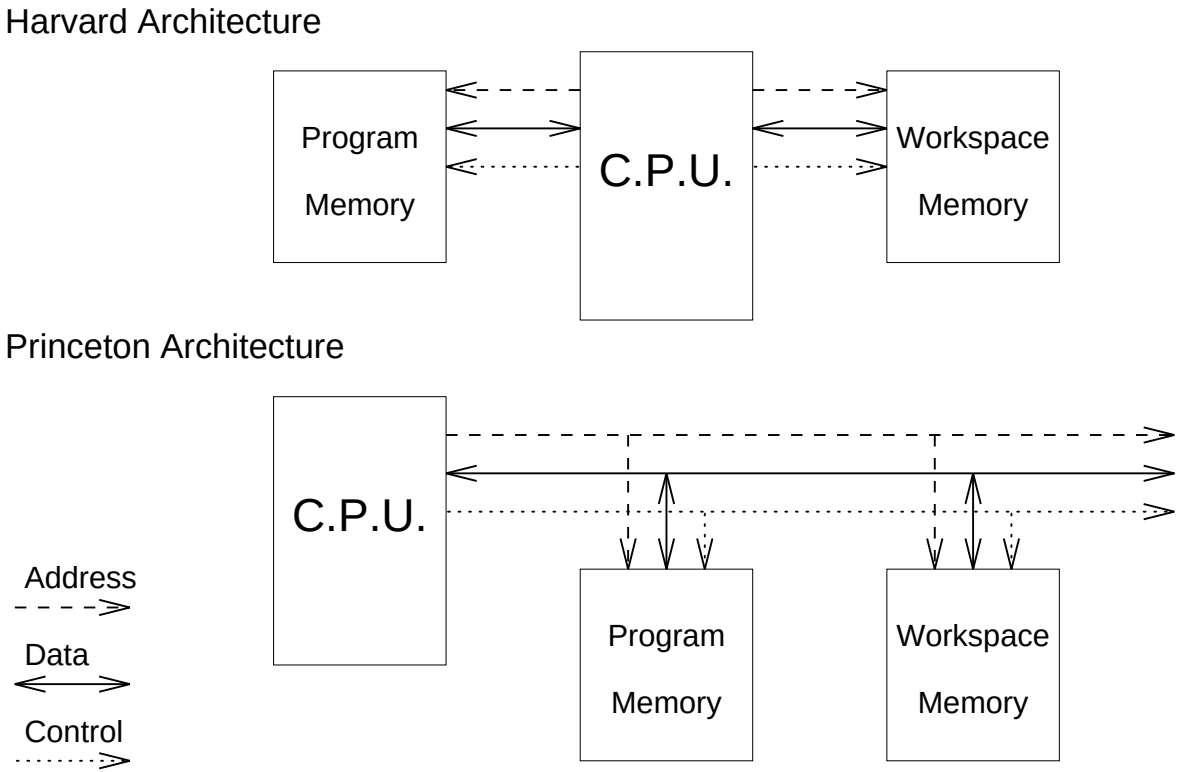
Review of Computer Architecture

- Central Processing Unit (C.P.U.)
  - Arithmetic Logic Unit (A.L.U.)
  - Instruction Processing Unit (I.P.U.)
  - Registers
- Support for Central Processing Unit

  Both the datapath and the control system require external storage.

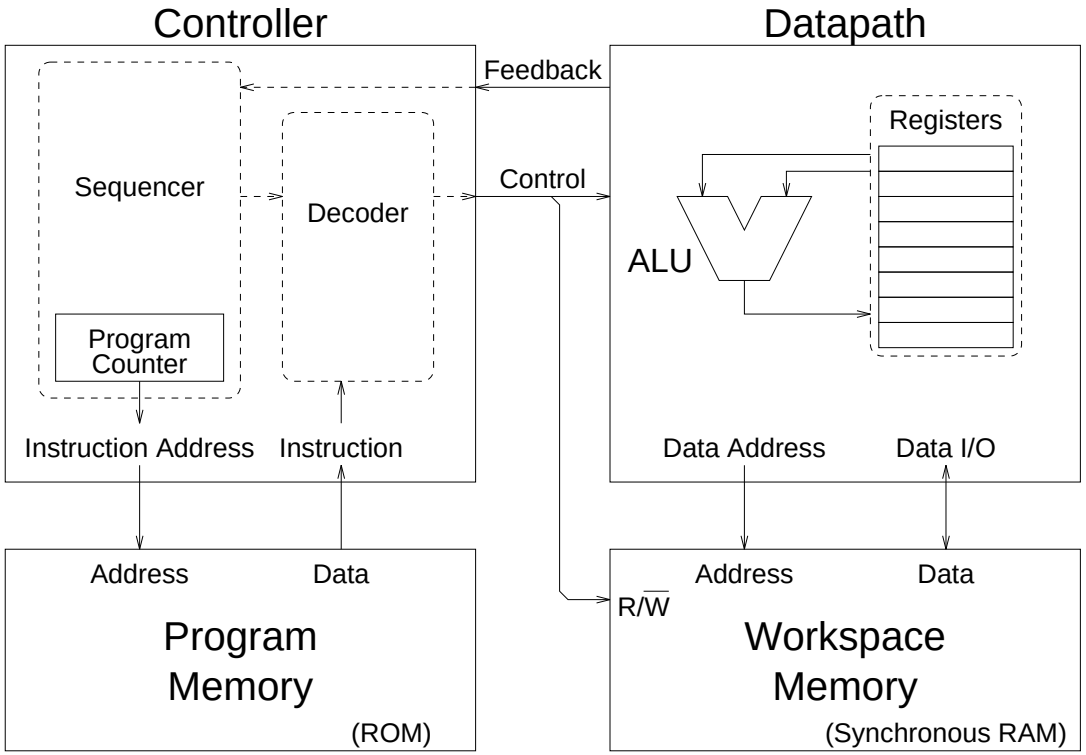


Review of Computer Architecture



- The Harvard Architecture uses different sets of buses for program and workspace memories.
- The Princeton Architecture uses a common set of buses for all components.

A Simple Processor Architecture



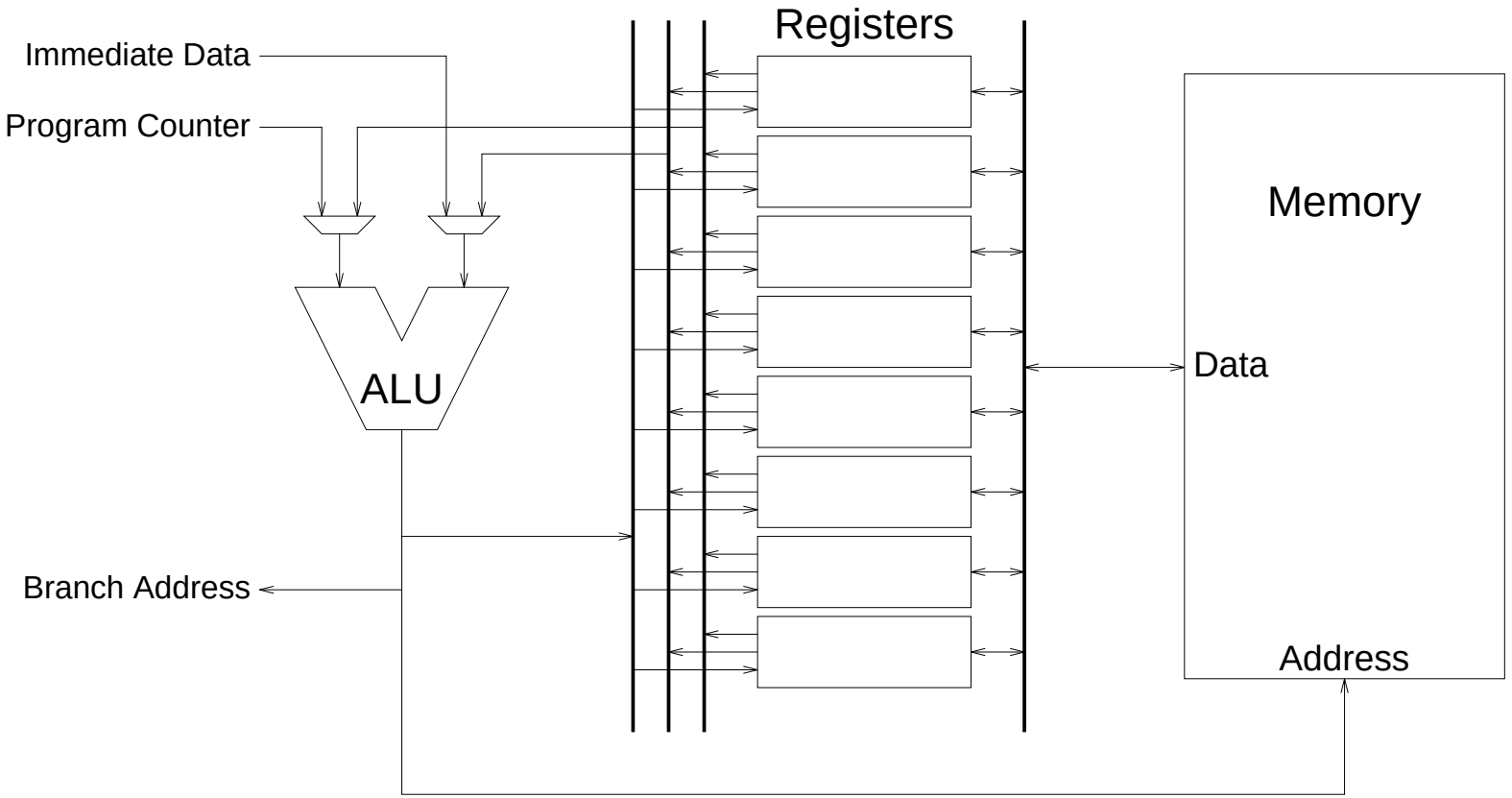
To introduce more advanced architectures we will consider a very simple processor with a Harvard type architecture. We will follow its refinement into a real processor and then we will base architecture advancements on this machine.

Instruction Types

We shall opt for a RISC style *register-register* architecture supporting three distinct instruction types:

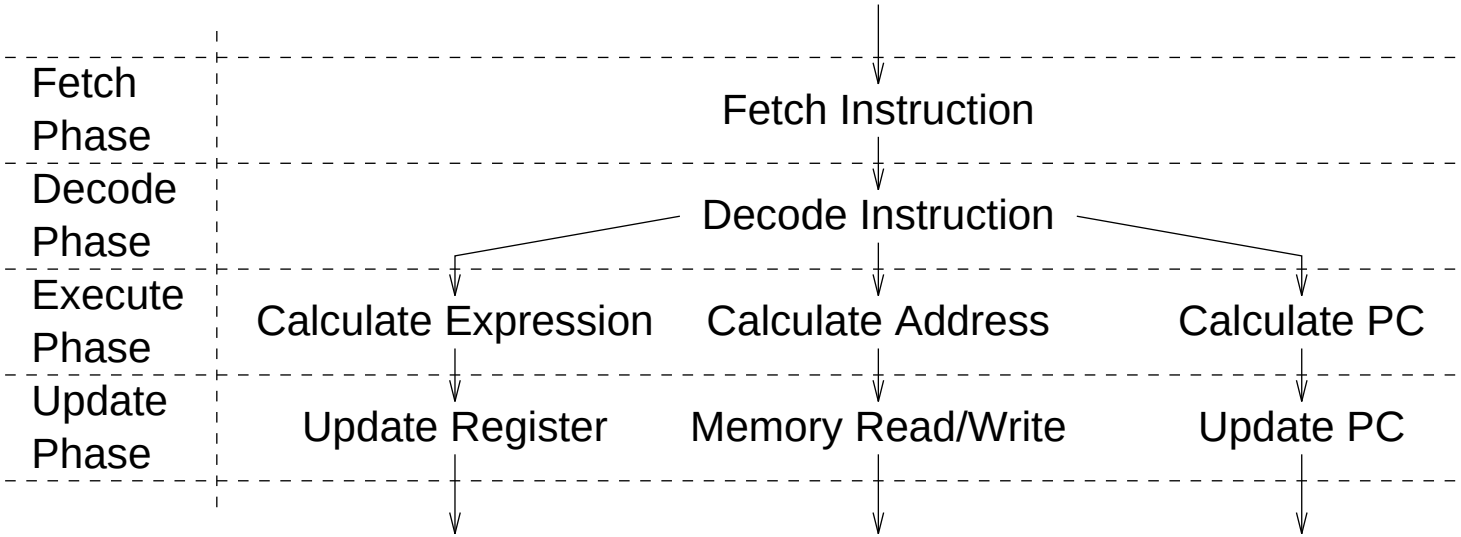
- Arithmetic/Logic instruction
  - The ALU is used for data processing.
  - No workspace memory access is made.
- Memory Access Instruction
  - The ALU is used for address calculation (e.g. simple indexing).
  - Workspace memory access may be read or write.
- Control Transfer Instruction
  - The ALU is used for program counter manipulation (e.g. conditional relative jump).
  - No workspace memory access is made.

Datapath Detail



- Additional paths allow data address and branch address calculation in the ALU
  - note that normal PC increment is still carried out by the sequencer
- Immediate data is available in all ALU modes

Instruction Cycle



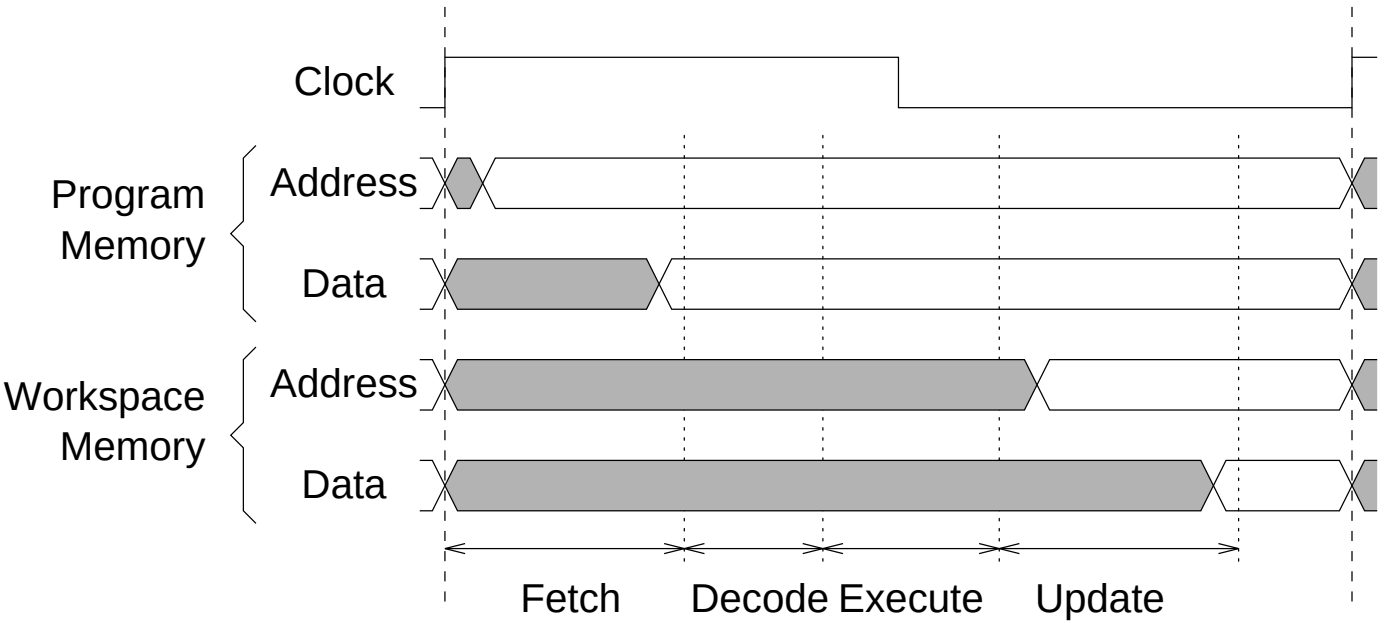
No unit is used more than once for each instruction[1], the whole instruction may be completed in a *single clock cycle*.

We have a **one CPI** machine[2].

[1] 1 program memory access, 1 instruction decode, 1 ALU calculation, 1 (or 0) data register update, 1 (or 0) workspace memory access

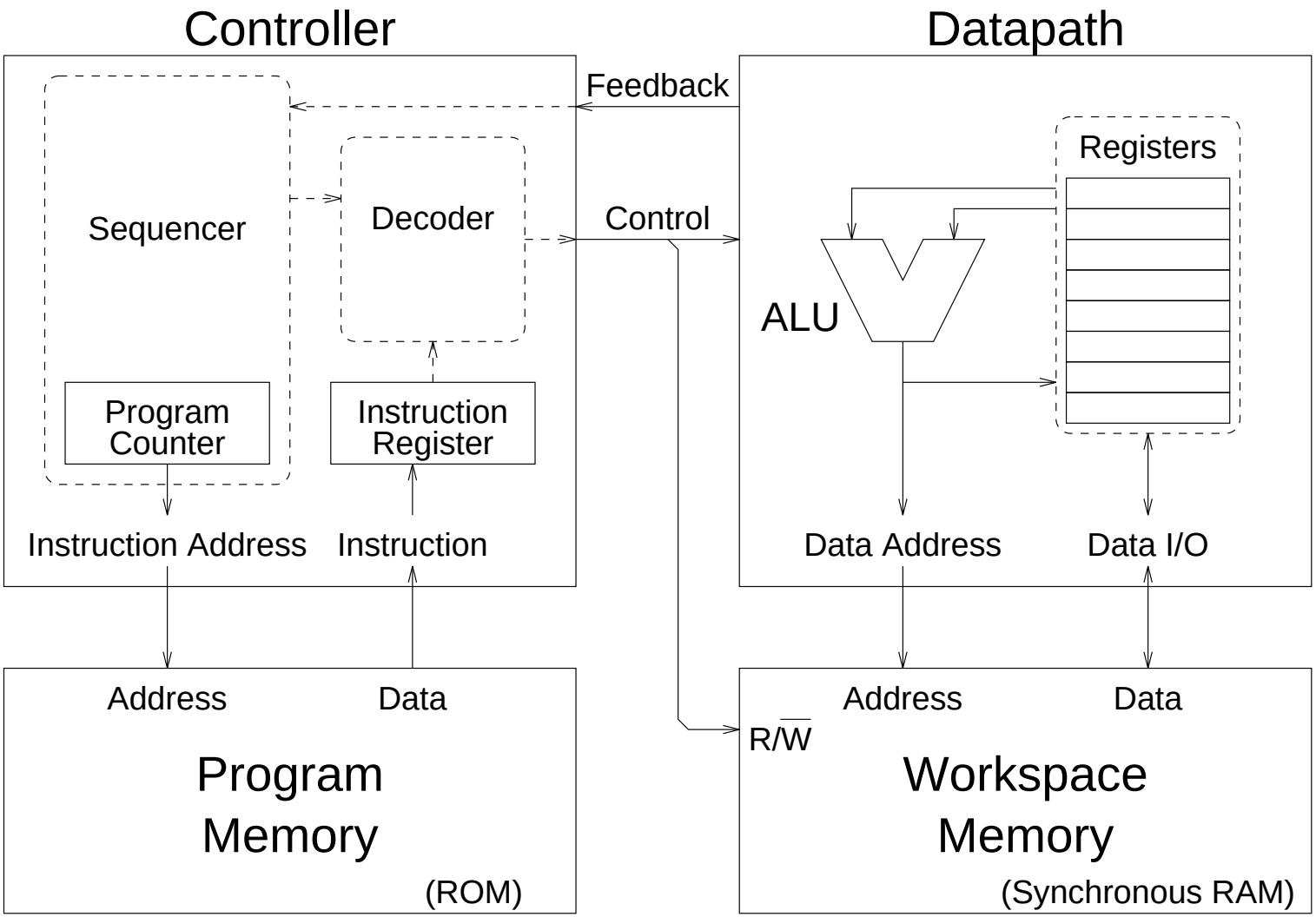
[2] **CPI** is **C**ycles **p**er **I**nstruction

Instruction Cycle



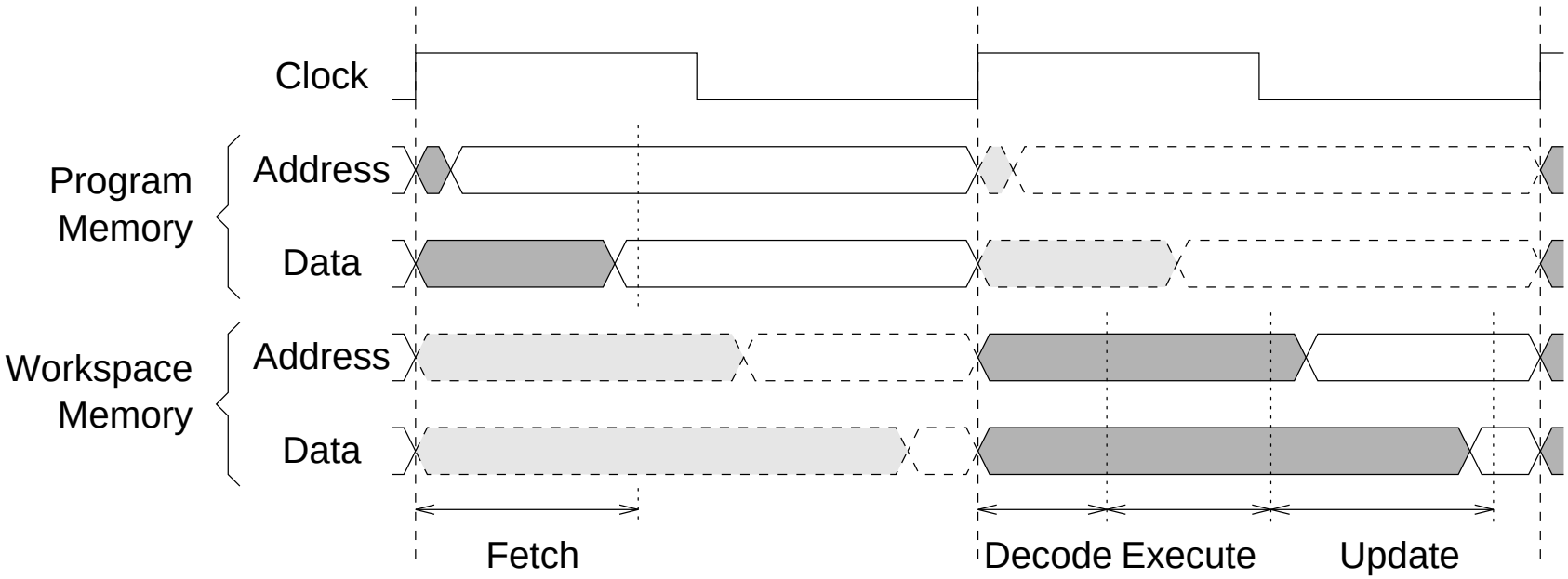
- The period of the clock must be long enough to accommodate the worst case timings for the worst case instruction.

Instruction Pre-Fetch



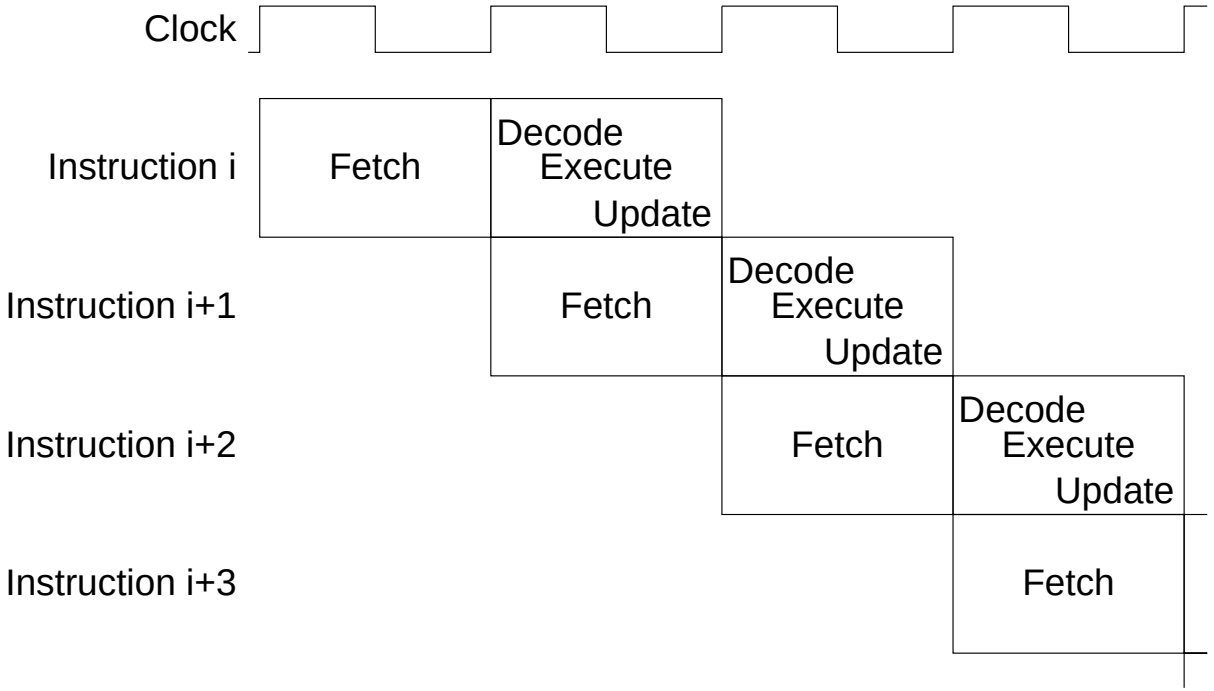
Instruction Pre-Fetch

By adding an instruction register we are able break the instruction cycle into two shorter cycles and allow instruction pre-fetch.



Instruction Pre-Fetch

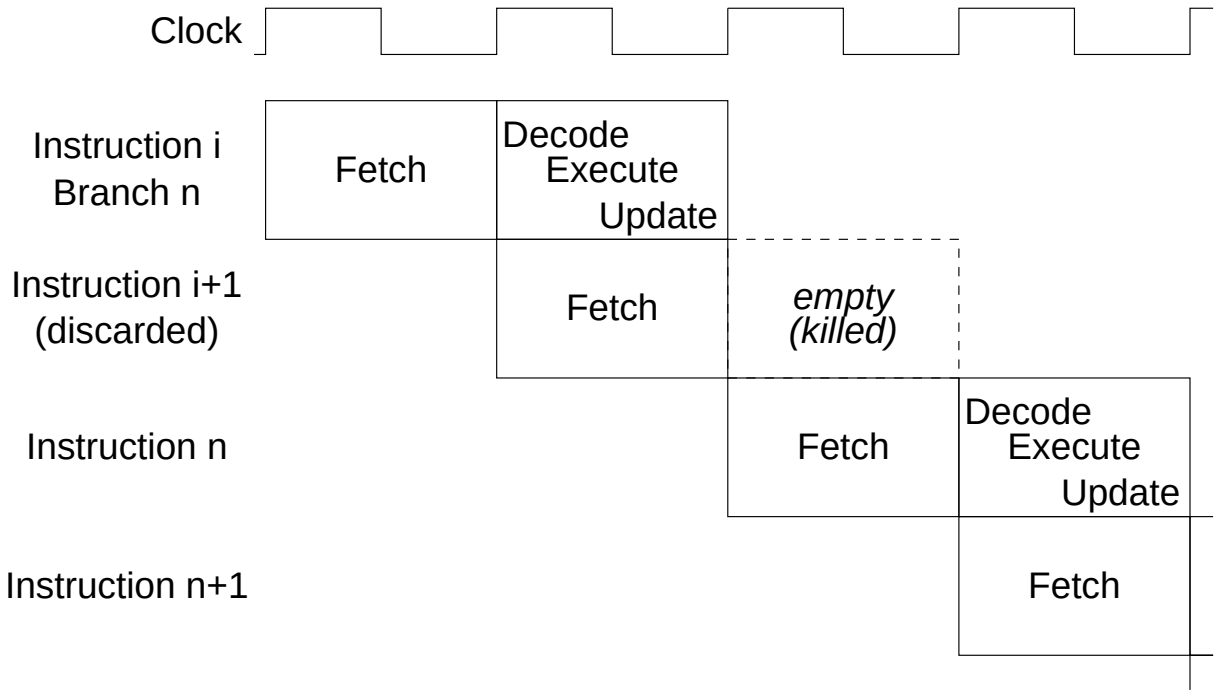
Instruction Pre-Fetch in a Harvard sequential computer:



- The next instruction is pre-fetched before an instruction is completed.

Instruction Pre-Fetch

- Following a Control Transfer Instruction we may pre-fetch the wrong instruction.

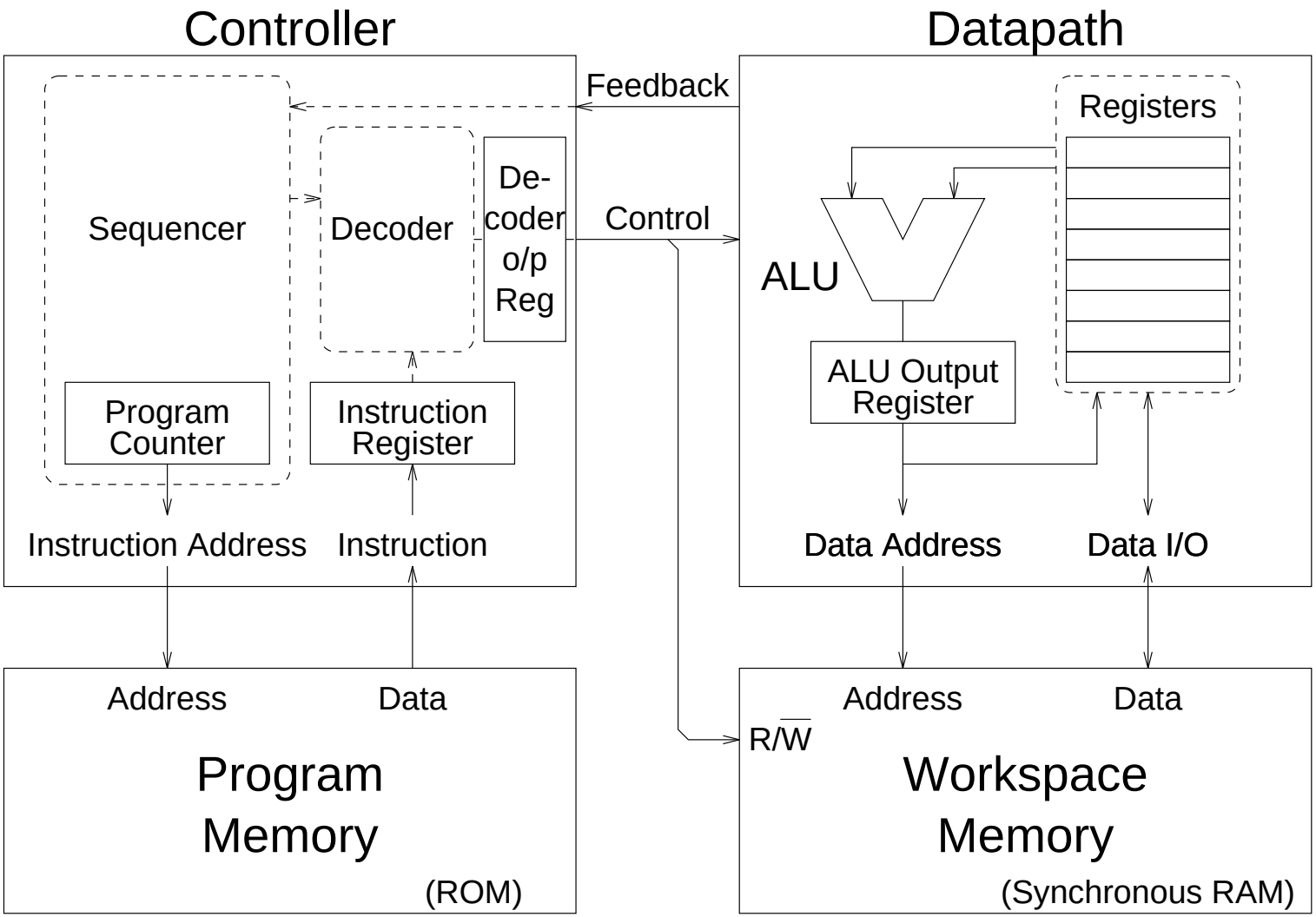


  - where the CTI is taken we must kill the instruction.
  - where the CTI is not taken we have already fetched the correct instruction.

Pipelining

- Instruction pre-fetch is a simple form of *pipelining*.
- We begin one operation before the previous operation has completed.
  - Although each individual instruction will take the same length of time to complete, a group of instructions may be completed in a much shorter time due to overlap.
  - The increase in performance arises since we are making more efficient use of our hardware. More of the hardware is kept busy for more of the time.
  - We have a limited form of *concurrency*
- A 4 CPI Harvard Architecture can be used to produce a 4 stage pipeline machine:

4 Stage Pipeline

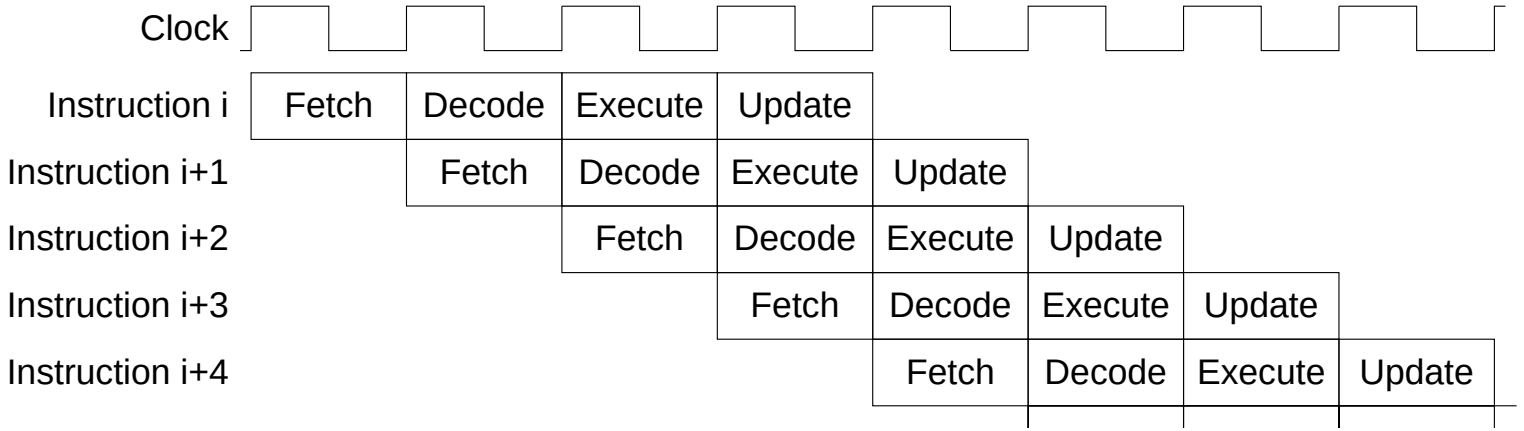


4 Stage Pipeline Operation

- Two more registers are added

  Decoder output register is updated at the end of cycle 2.

  ALU output register is updated at the end of cycle 3.



- A new instruction is fetched on every clock cycle.
- The machine completes four quarter instructions every clock cycle giving an average of one CPI.

Processor Performance

- Let us consider the performance of our simple machines:

$$Work\ Rate = \frac{Instruction\ Rate}{Instructions\ per\ Task}$$

  - The *Instructions per Task* value will be determined by the power of our instruction set and how well it matches the task to be performed.

$$Instruction\ Rate\ (Mips) = \frac{Clock\ Frequency\ (MHz)}{Cycles\ per\ Instruction}$$

  - The *Cycles per Instruction* value is an average over the whole task. At first it appears that all three architectures will have a CPI value of one, allowing them to be compared on clock frequency alone. In fact the average CPI values for the two pipelined machines will be greater than one due to problems in the pipeline operation.

Abuse of Statistics

- Mips[3]
  - The *Instruction Rate* of a machine is frequently given as a measure of its performance. Since an average *Instruction Rate* may depend on the set of instructions chosen the maximum *Instruction Rate* is quoted.
    Here we can take *Mips* to mean *Meaningless Indication of Processor Speed*, the best we can hope for is an indication of the relative performance of machines sharing the same instruction set and a similar architecture.
- VAX Mips
  - Relative Mips value w.r.t. to VAX 11/780.
    This is better but will still dependent on the task chosen for the comparison.
- Benchmarking
  Architects & Compiler Writers *vs* Benchmark Designers.
  - Alpha 21264 (@833 MHz) 544 SPECint2000 658 SPECfp2000 (3332_{max} Mips).
  - Intel Pentium 4 (@1.5 GHz) 535 SPECint2000 561 SPECfp2000 (1500_{max} Mips).

[3]Millions of Instructions Per Second