Pipeline Design

Vocabulary:

The depth of a pipeline is the number of stages it has.

A shallow pipeline has few stages.

A deep pipeline has many stages.

A timing template gives the generic layout of pipeline stages.

Fundamental Constraint of pipelined execution: The observed execution must be functionally equivalent to executing the instructions one-at-a-time.

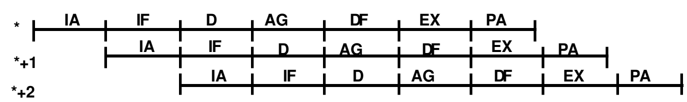
Timing Template

Pipeline design begins by roughing out a timing template and assigning activities to the stages.

The tasks of instruction interpretation can obviously be partitioned into stages in a variety of ways, so there is no consistency in chosing or naming them.

IA = Instruction Address Generation
IF = Instructin Fetch
D = Decode
AG = Address Generation

DF = Data Fetch
EX = Execution
PA = Put Away



Pipeline Design Classes

A *static pipeline* design requires each instruction to transit each stage even if the instruction does not require the operations of that stage.

Dynamic pipelines allow instructions to by-pass one or more stages if the operations of the stages are not needed.

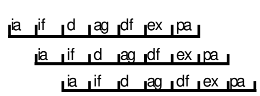
In dynamic pipelining "by-passing" is interpreted aggressively.

Type 1: Instructions are decoded in the order specified by the program, and the values are "put away" in order.

If later instructions execute before eariler instructions, the results are held, and only put away in sequence.

Variant: Relax putaway to keep memory write-back in order, but allow register write-back out of order.

```
ST.W    R4,SomeLoc     ; Save away result
SUB.W   R5,R5          ;  Now zero accum
LD.W    R6,OtherLoc    ;  Get another word
```



Pipeline Classes Continued

Type 2: Instructions are decoded in the order specified by the program, but they are executed in any order that is consistent with dependencies on earlier instructions.

Type 3: Instructions are issued in any order subject only to the "Fundmenal Constraint" and they are executed as soon as the instructions on which they depend (if any) complete.

Example 3033

IBM 3033 -- An old (early 80's) mainframe, pre RISC.

- 2 cycle cache access
- Decode combined with address generation
- IA included here, but usually ignored
- Cycle time = 57ns
- Dynamic (type 1) pipeline
- PA by-pass to DA available

```
ia if if da df df ex pa
   ia if if da df df ex pa
      ia if if da df df ex pa
```


Example V8

Amdahl 470 V/8, a more contemporary "360" ISA

- R, register access, C, error check, W, write
- Two-phase clocking
- Issue rate, 1 instruction per cycle pair
- Cycle time = 26ns

```
ia if if d r ag df df ex ex c w
      ia if if d r ag df df ex ex c w
            ia if if d r ag df df ex ex c w
```