Pipelining

In a computer, the instruction cycle can be broken into smaller, shorter tasks. This is known as pipelining. The main motivation to do so is to increase the rate at which instructions are executed. However, we will see that there are some effects that the programmer needs to be aware of.

Before examining the pipeline in a CPU, we will look at a simpler example.

Introduction to Pipelining – Car Manufacturing Plant

ONE TEAM

Chassis

Engine

Hood, Doors

Paint

Wheels

FIVE TEAMS (unionized labor)

Chassis	Engine	Hoods,Door	Paint	Wheels

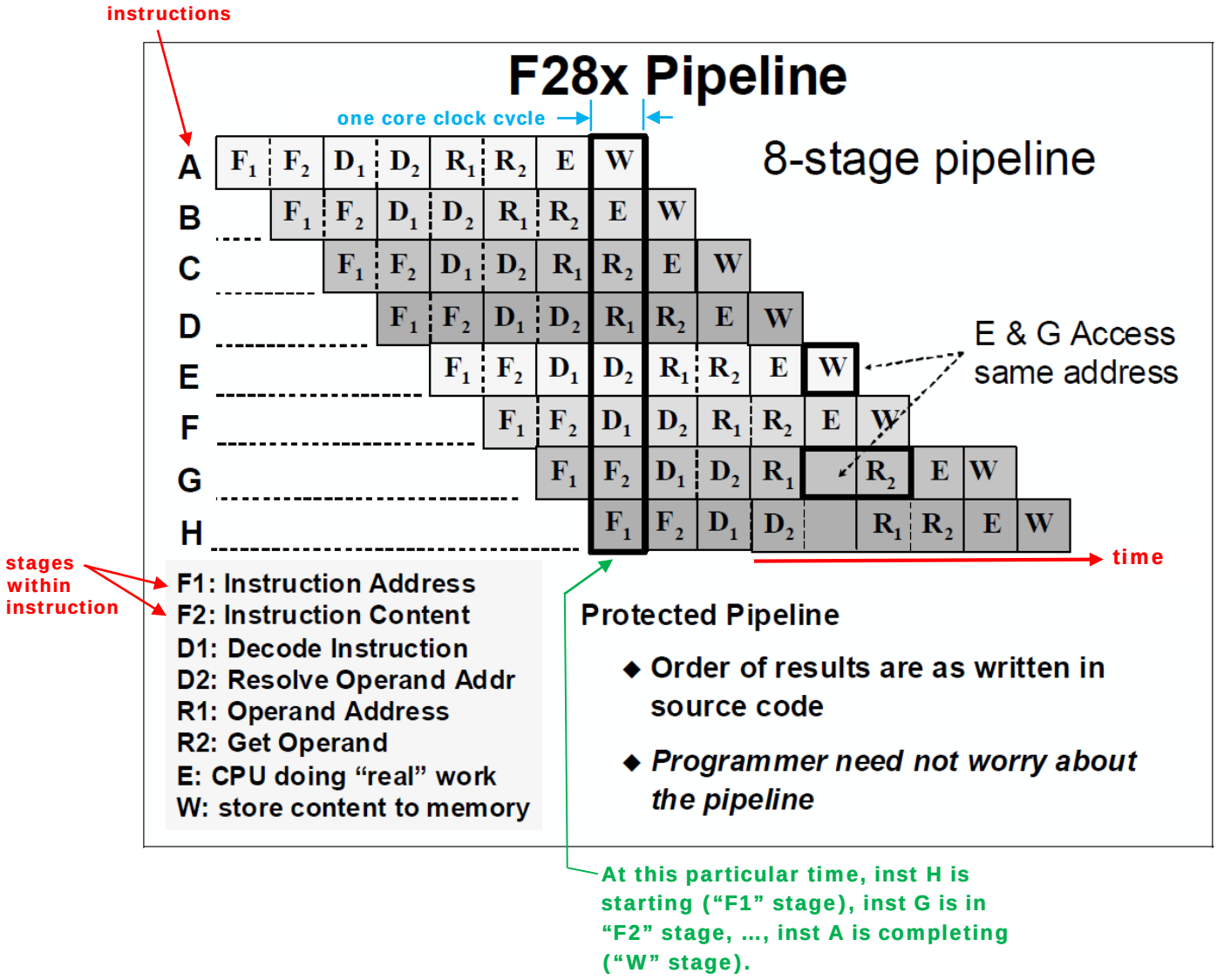
FIVE TEAMS (non-unionized labor)

Chassis	Engine	Hoods,Door	Paint	Wheels

What is a possible cause of a pipeline “stall”?

What is a possible cause of a pipeline “flush”?

Pipelining in c2000 CPU



Note that the Harvard architecture allows instructions to be fetched at the same time operations are performed on the data.

If the core clock = 60 MHz…

1. How long does it take an instruction to complete from start to finish?

2. When the pipeline is "full", at what rate do instructions complete?

The pipeline can encounter conditions that cause it to stall or flush.

e.g. of pipeline stall (i.e., slowdown):

```
y = 1;
b = 128;
x = 2;
.
.
.
x = y + 3;  //x is written to in inst E's "W" stage
a = b << 6;
z = x + 5;  //x is read in inst G's "R2" stage
```

example of read-modify-write with conflict

"protected" pipeline automatically prevents a "dirty" read

e.g. of pipeline flush (i.e., re-start):

e.g. of pipeline flush (i.e., re-start):

Stalls and flushes cause the instruction throughput to degrade.

##########

Another factor for the programmer to consider is latency in activation of GPIO outputs and response to GPIO inputs.