

DYNAMIC SCHEDULING

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

University of Utah

Overview

- Announcement

- ▣ Homework 3 will be uploaded tonight

- This lecture

- ▣ Dynamic scheduling

- Forming data flow graph on the fly

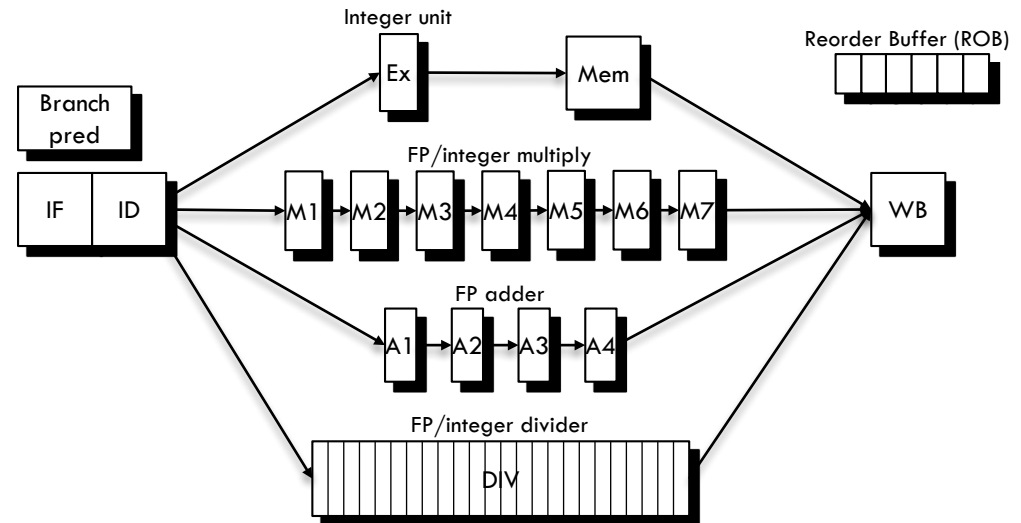
- ▣ Register renaming

- Removing false data dependence

- Architectural vs. physical registers

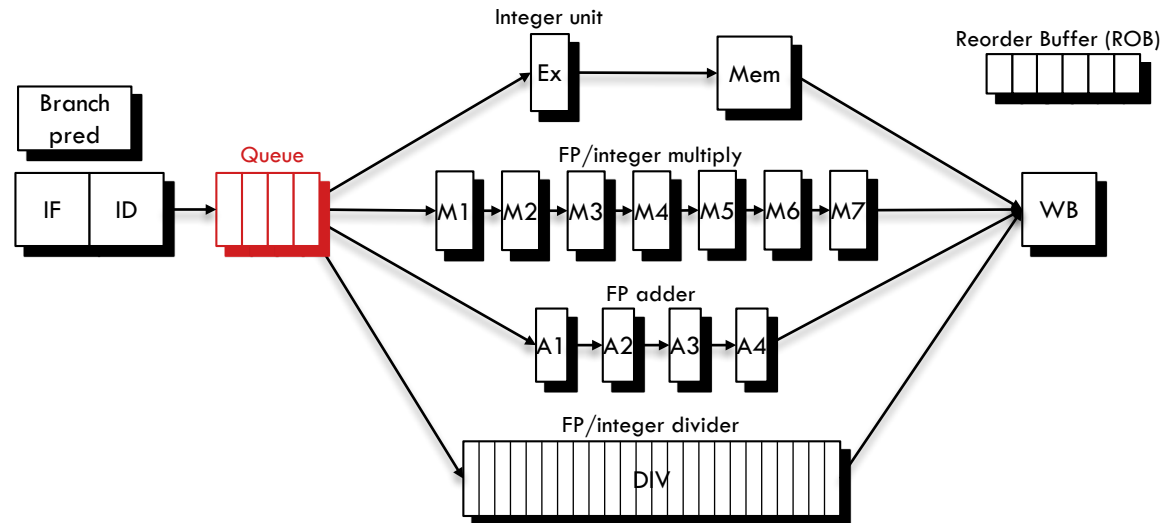
Big Picture

- **Goal:** exploiting more ILP by avoiding stall cycles
 - ▣ Branch prediction can avoid the stall cycles in the frontend



Big Picture

- **Goal:** exploiting more ILP by avoiding stall cycles
 - ▣ Branch prediction can avoid the stall cycles in the frontend
 - **More instructions are sent to the pipeline**



Big Picture

- **Goal:** exploiting more ILP by avoiding stall cycles
 - ▣ Branch prediction can avoid the stall cycles in the frontend
 - More instructions are sent to the pipeline
 - ▣ Instruction scheduling can remove unnecessary stall cycles in the execution/memory stage
 - **Static scheduling**
 - Complex software (compiler)
 - Unable to resolve all data hazards (no access to runtime details)
 - **Dynamic scheduling**
 - Completely done in hardware

Dynamic Scheduling

- **Key idea:** creating an instruction schedule based on runtime information
 - ▣ Hardware managed instruction reordering

Assembly code:

```
DIV F1, F2, F3
```

```
ADD F4, F1, F5
```

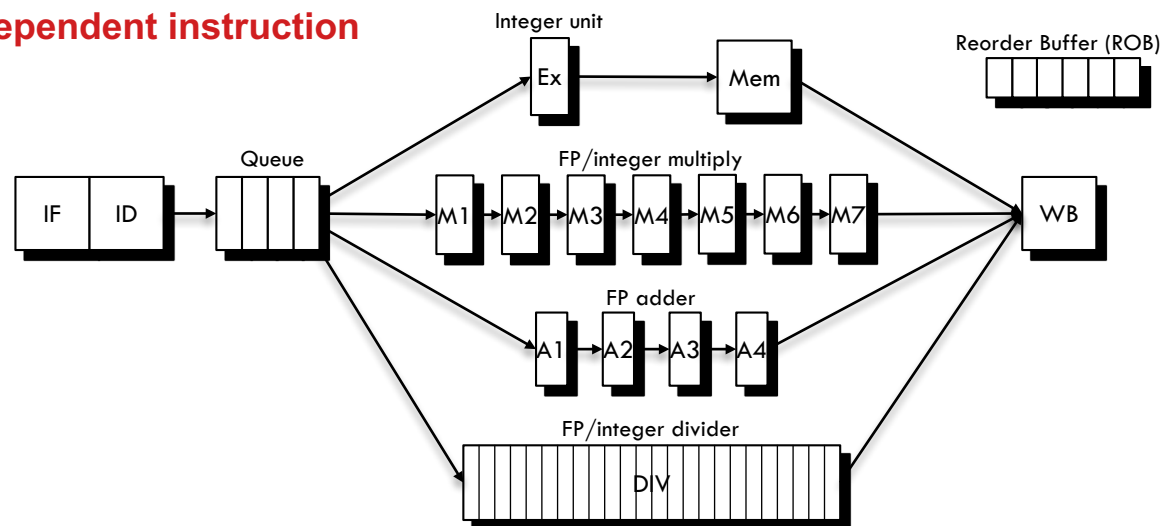
```
SUB F6, F5, F7
```

Long latency operation

Dependent instruction

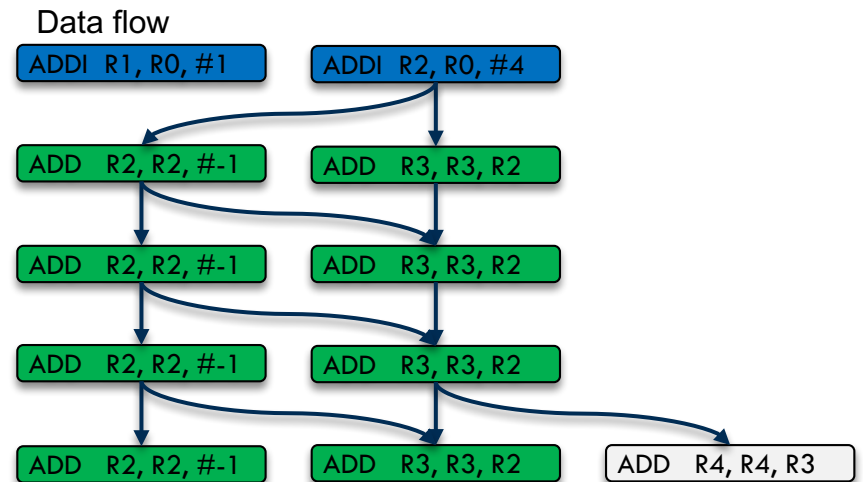
Independent instruction

Out-of-order execution?



Dynamic Scheduling

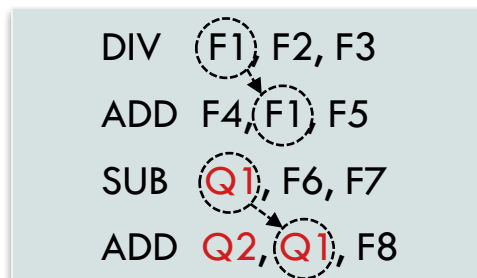
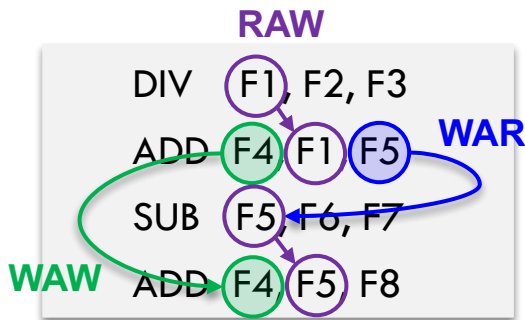
- **Key idea:** creating an instruction schedule based on runtime information
 - ▣ Hardware managed instruction reordering
 - ▣ Instructions are executed in data flow order



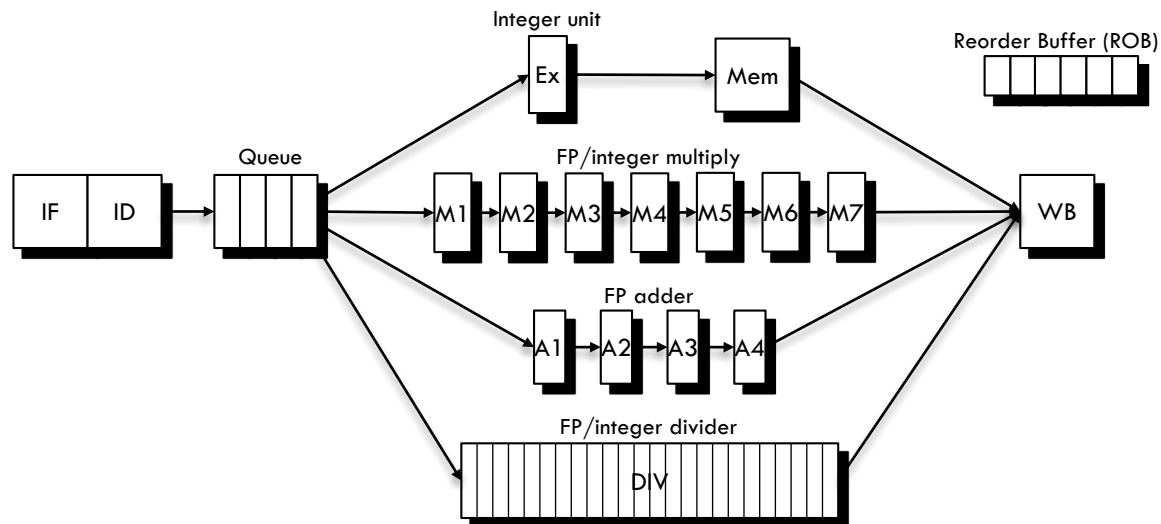
How to form data flow graph on the fly?

Register Renaming

- Eliminating WAR and WAW hazards
 - ▣ Change the mapping between architectural registers and physical storage locations



WAR and WAW hazards can be removed using more registers

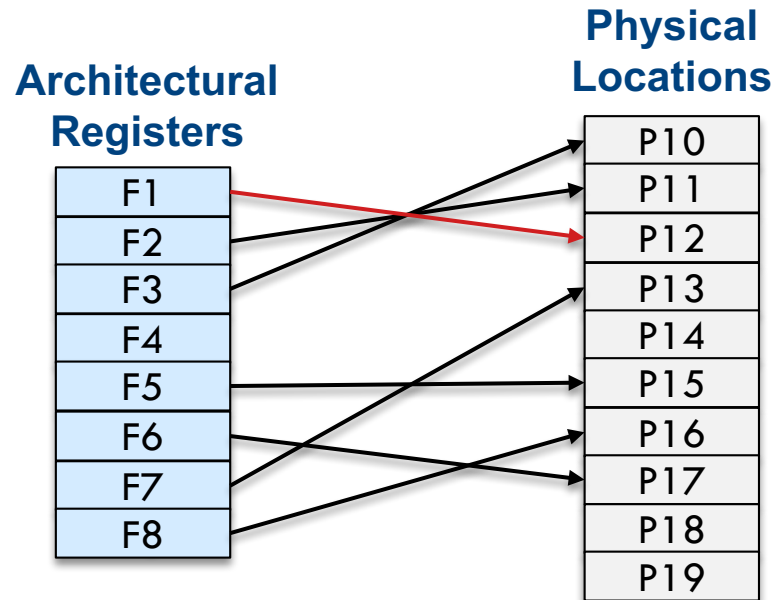


Register Renaming

- Eliminating WAR and WAW hazards
 - 1. allocate a free physical location for the new register
 - 2. find the most recently allocated location for the register

```
DIV  F1, F2, F3
ADD  F4, F1, F5
SUB  F5, F6, F7
ADD  F4, F5, F8
```

```
DIV  P12, P11, P10
```

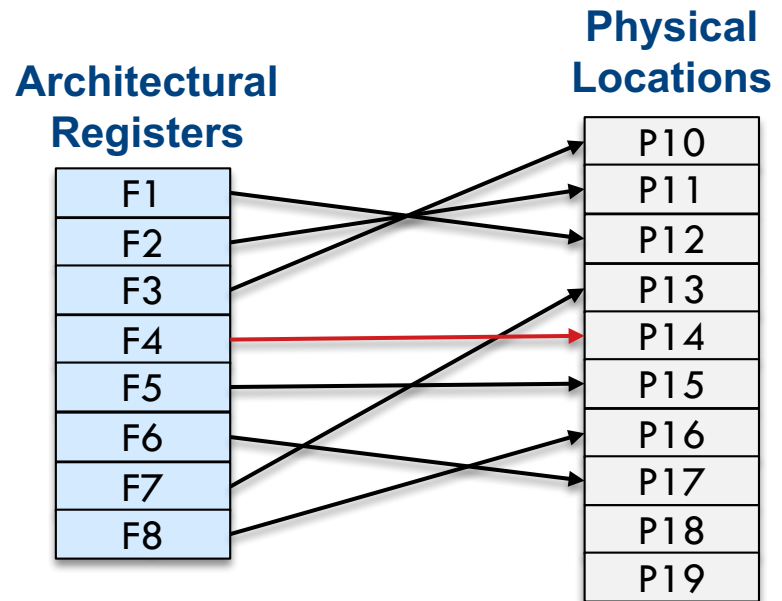


Register Renaming

- Eliminating WAR and WAW hazards
 - 1. allocate a free physical location for the new register
 - 2. find the most recently allocated location for the register

```
DIV  F1, F2, F3
ADD  F4, F1, F5
SUB  F5, F6, F7
ADD  F4, F5, F8
```

```
DIV  P12, P11, P10
ADD  P14, P12, P15
```

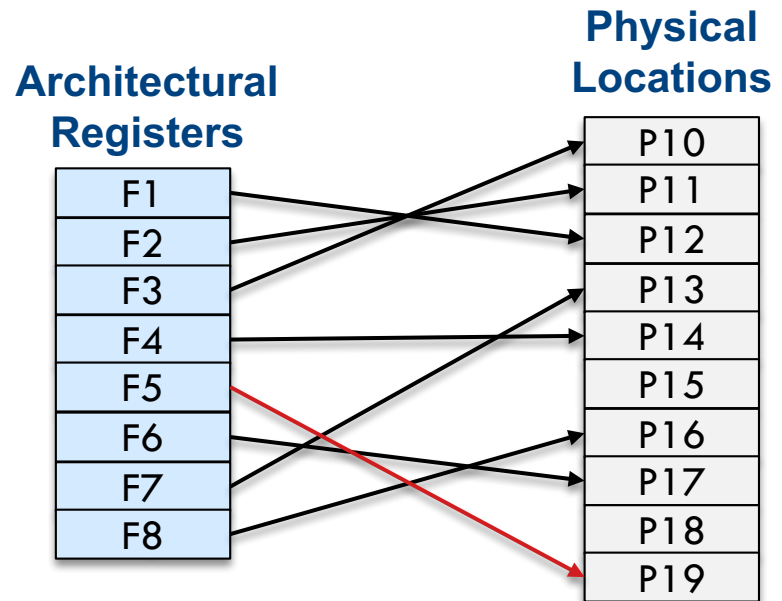


Register Renaming

- Eliminating WAR and WAW hazards
 - 1. allocate a free physical location for the new register
 - 2. find the most recently allocated location for the register

```
DIV  F1, F2, F3
ADD  F4, F1, F5
SUB   F5, F6, F7
ADD  F4, F5, F8
```

```
DIV  P12, P11, P10
ADD  P14, P12, P15
SUB  P19, P17, P13
```



Register Renaming

- Eliminating WAR and WAW hazards
 - 1. allocate a free physical location for the new register
 - 2. find the most recently allocated location for the register

```
DIV  F1, F2, F3
ADD  F4, F1, F5
SUB   F5, F6, F7
ADD  F4, F5, F8
```

```
DIV  P12, P11, P10
ADD  P14, P12, P15
SUB   P19, P17, P13
ADD  P18, P19, P16
```

