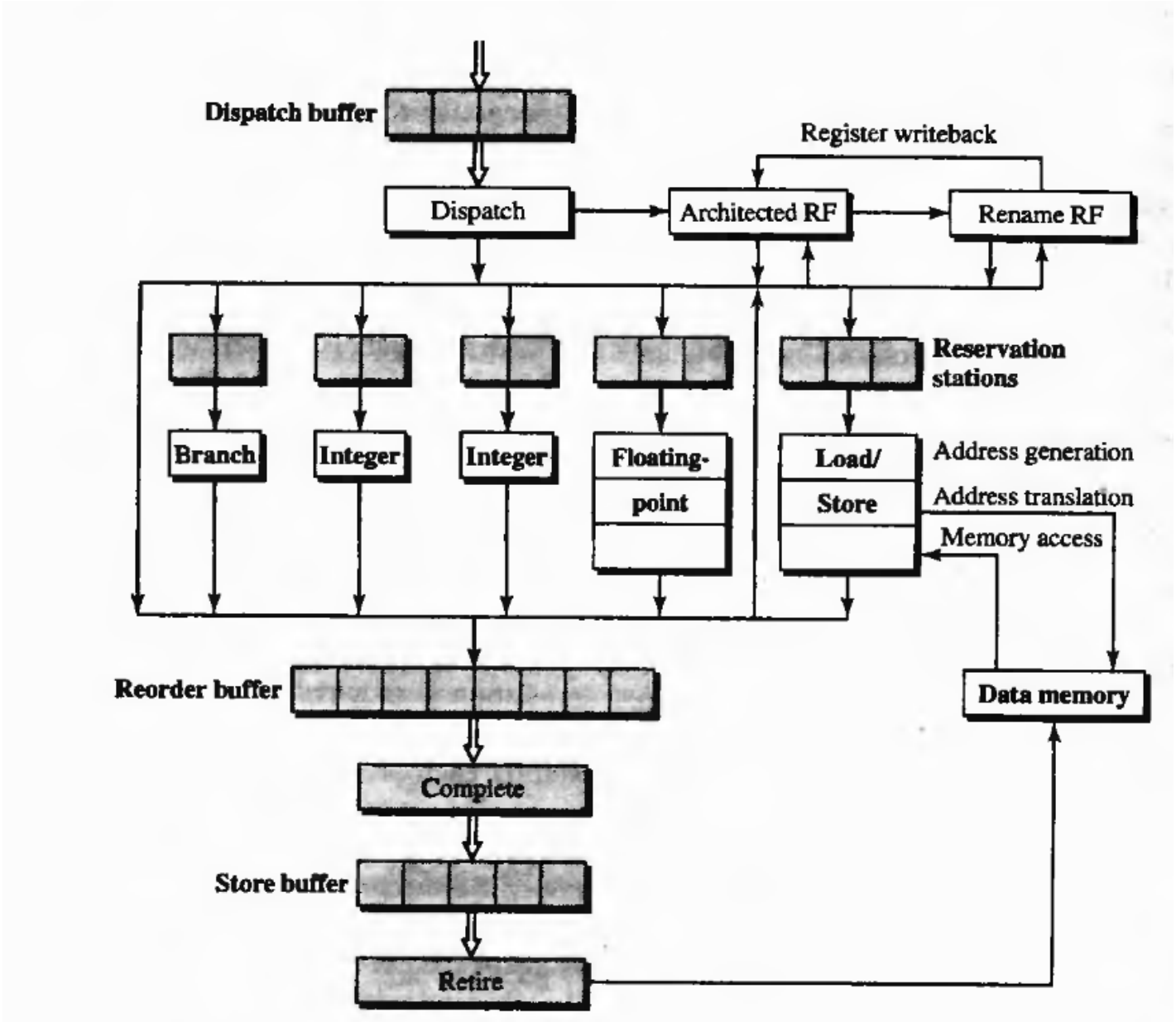
Out-of-Order Memory Access

Mateusz Bajorski

November 12, 2014

In order memory access



What is the problem?

What is the problem?

1. Store or Load is blocking pipe for futher Load/Store

Overview

Need to check what we need and when we need to get operation done

1. First stage - generate address

Pipeline stages

1. First stage - generate address
2. Second stage - translate address

Pipeline stages

1. First stage - generate address
2. Second stage - translate address
3. Third stage - get data

Load

What we need to know

Load

What we need to know

1. Register operand

Store

What we need to know

Store

What we need to know

1. Register operand

Store

What we need to know

1. Register operand
2. Data operand

Store problems

Store problems

1. Is that saved?

Store problems

1. Is that saved?
2. When it will be saved?

Store problems

1. Is that saved?
2. When it will be saved?
3. What if we have to rollback?

Store problems

1. Is that saved?
2. When it will be saved?
3. What if we have to rollback?

Example

$$Y(i) = A * X(i) + Y(i) \tag{1}$$

```
 1       F0 <- LD,a
 2       R4 <- ADDI,Rx,#512  ;last address
 3 Loop:
 4       F2 <- LD,0(Rx)      ;load X(i)
 5       F2 <- MULTD,F0,F2   ;A*X(i)
 6       F4 <- LD,0(Ry)      ;load Y(i)
 7       F4 <- ADDF,F2,F4    ;A*X(i)+Y(i)
 8       0(Ry) <- SD,F4      ;store into Y(i)
 9       Rx <- ADDI,Rx,#8    ;inc. index to X
10       Ry <- ADDI,Ry,#8    ;inc. index to Y
11       R20 <- SUB,R4,Rx    ;compute bound
12       BNZ,R20,Loop        ;check if done
13
14
```

Data dependencies

Data dependencies

1. RAW - Read After Write

Data dependencies

1. RAW - Read After Write
2. WAR - Write After Read

Data dependencies

1. RAW - Read After Write
2. WAR - Write After Read
3. WAW - Write After Write

Data dependencies

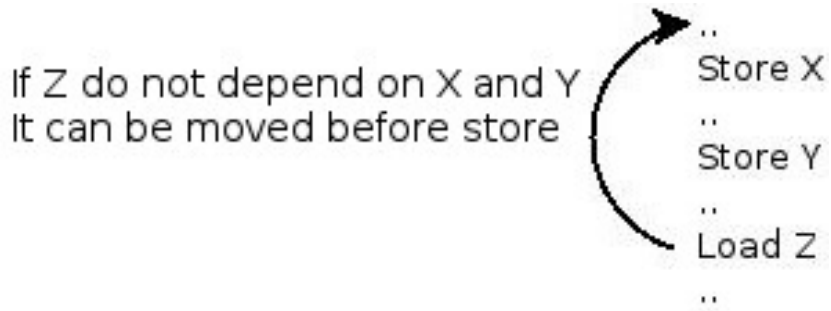
1. RAW - Read After Write
2. WAR - Write After Read
3. WAW - Write After Write

How we can improve this?

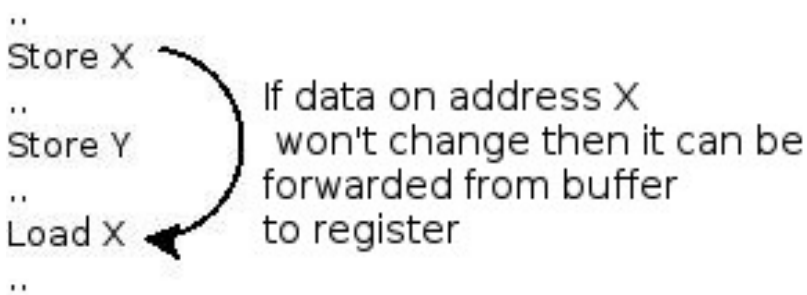
```
 1       F0 <- LD,a
 2       R4 <- ADDI,Rx,#512  ;last address
 3 Loop:
 4       F2 <- LD,0(Rx)      ;load X(i)
 5       F2 <- MULTD,F0,F2   ;A*X(i)
 6       F4 <- LD,0(Ry)      ;load Y(i)
 7       F4 <- ADDF,F2,F4    ;A*X(i)+Y(i)
 8       0(Ry) <- SD,F4      ;store into Y(i)
 9       Rx <- ADDI,Rx,#8    ;inc. index to X
10       Ry <- ADDI,Ry,#8    ;inc. index to Y
11       R20 <- SUB,R4,Rx    ;compute bound
12       BNZ,R20,Loop        ;check if done
13
14
```

Starting fixing

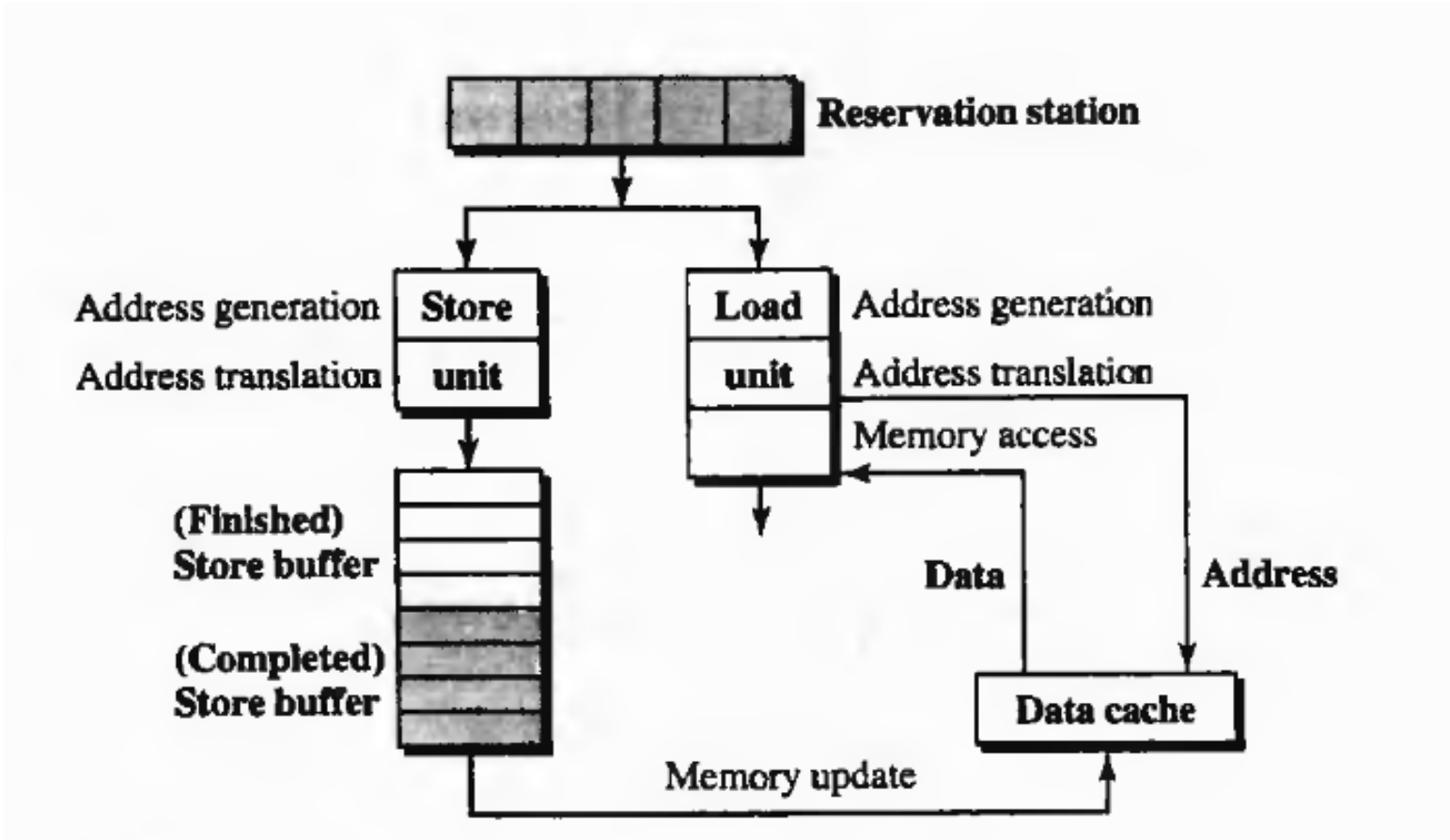
Load bypassing



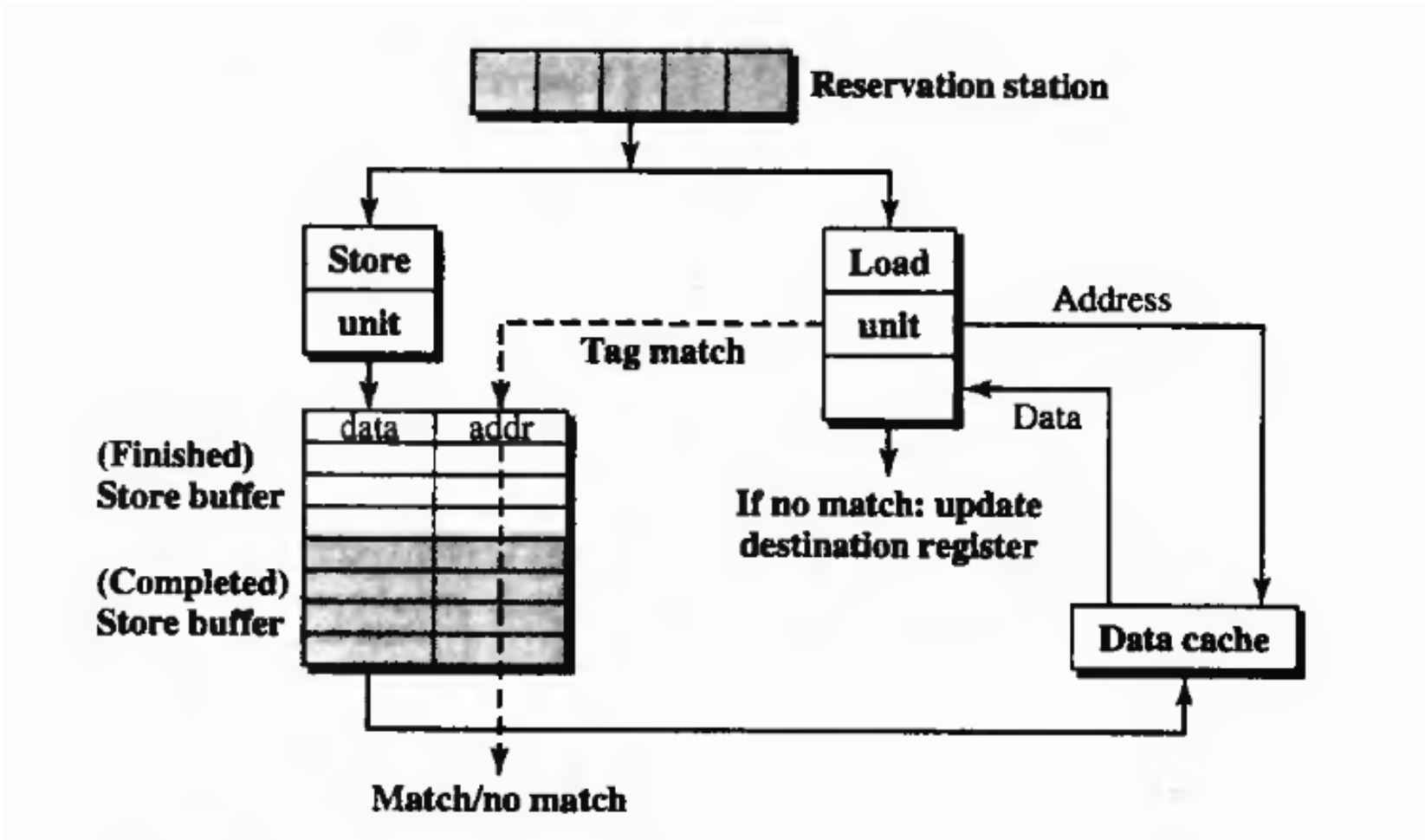
Load forwarding



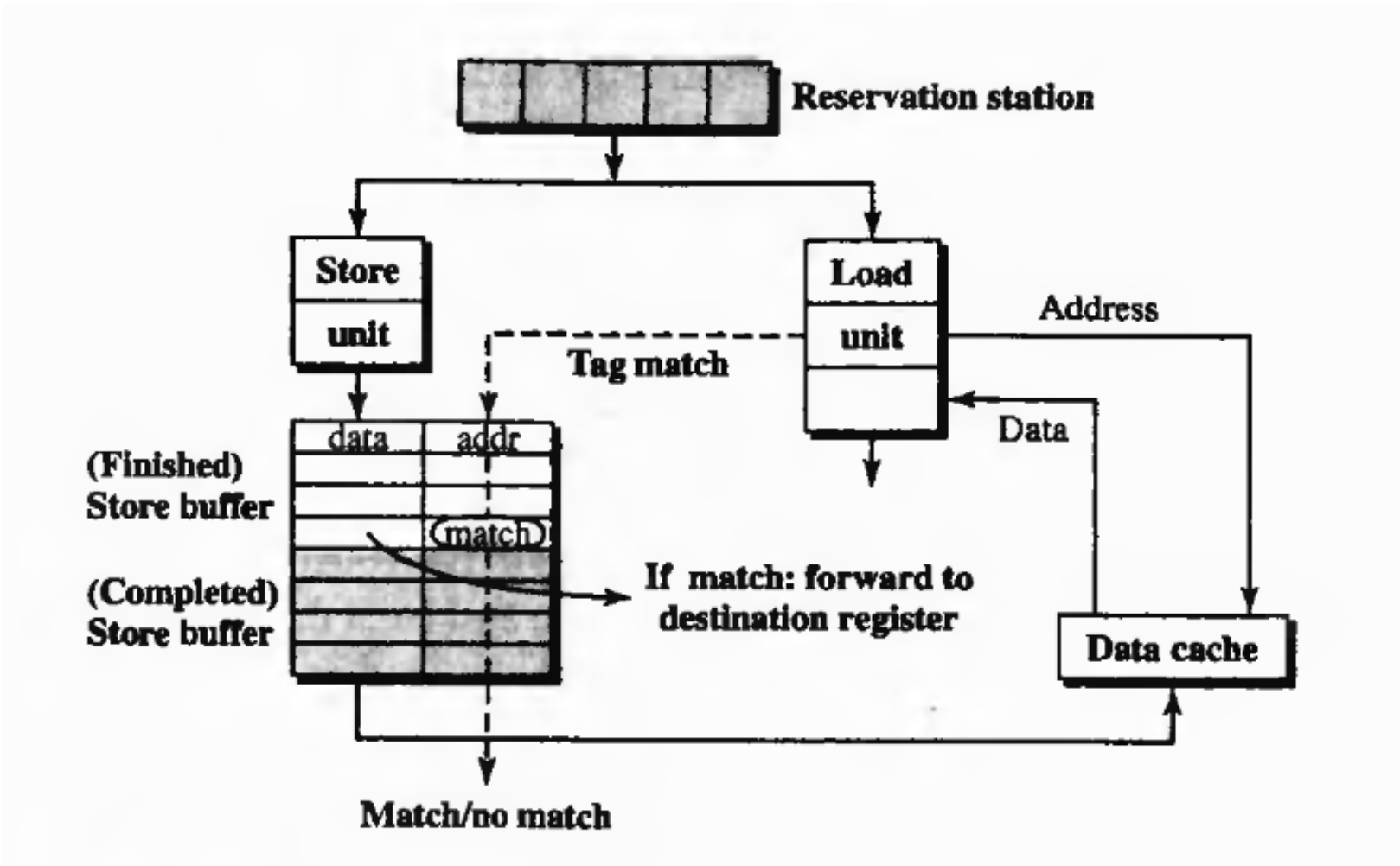
load/store mechanism with store buffer



Implementing load bypassing



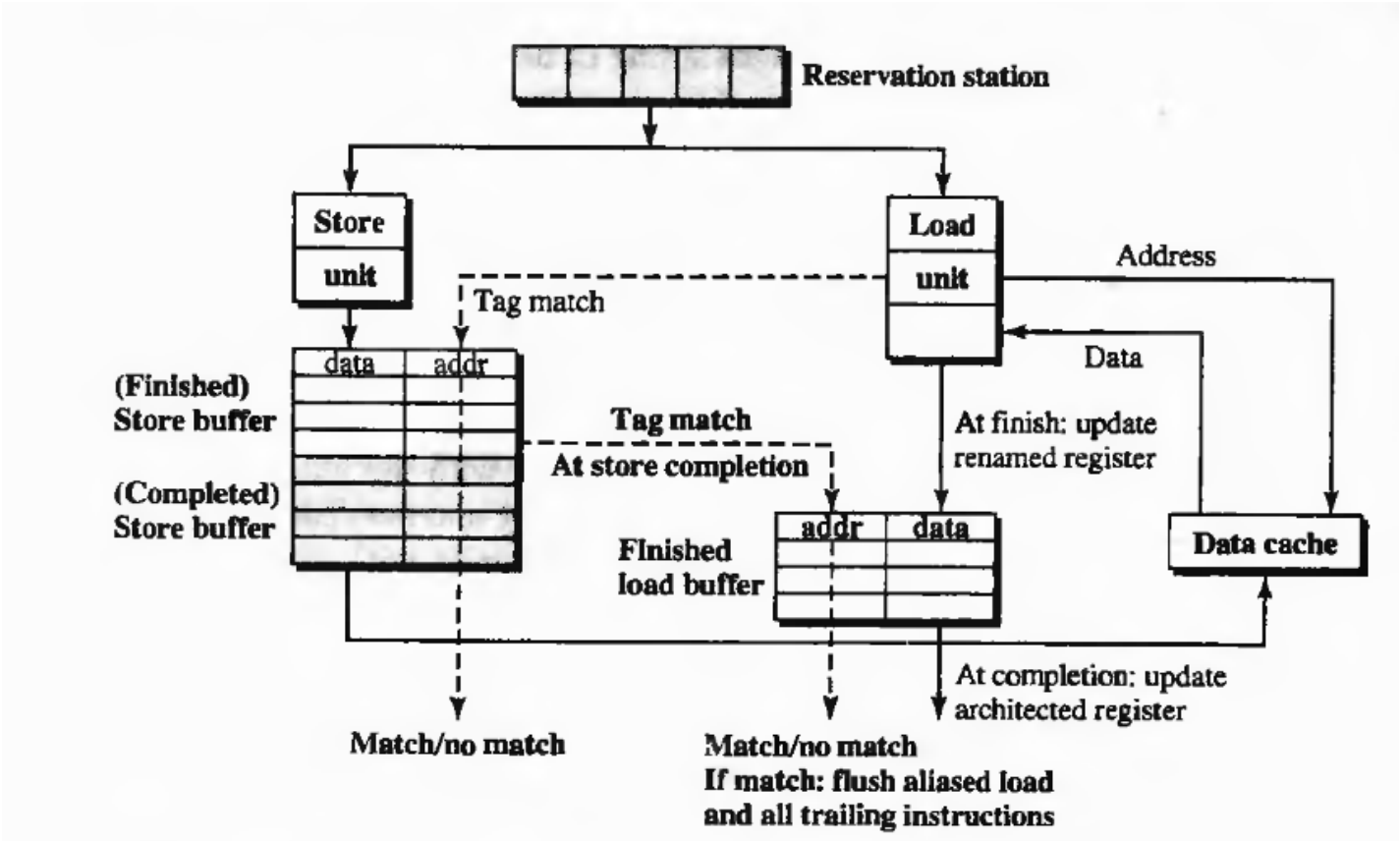
Implementing load forwarding



What if?

What if load could finish before store operation on the same address?

Finished load buffer

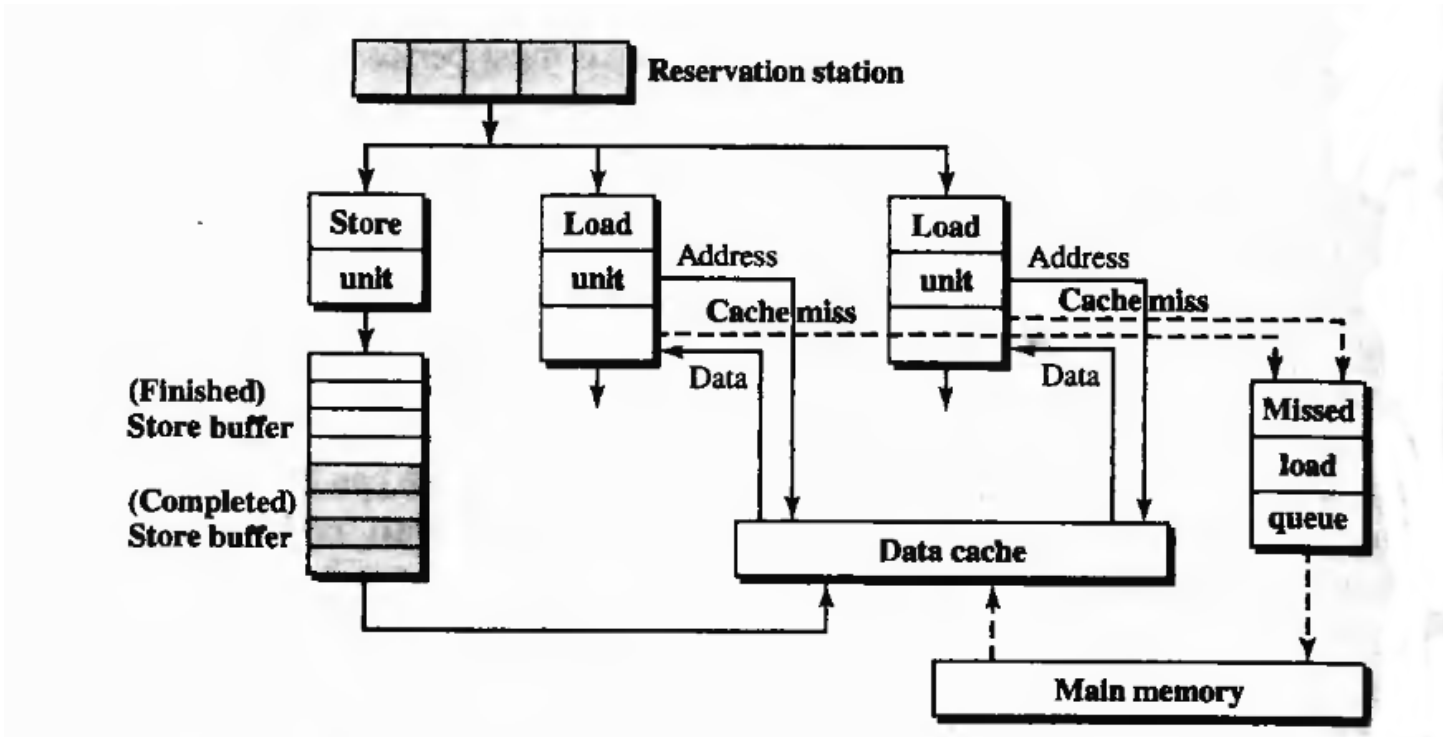


This isn't even my final form

How We can try to reduce or avoid miss penalty?

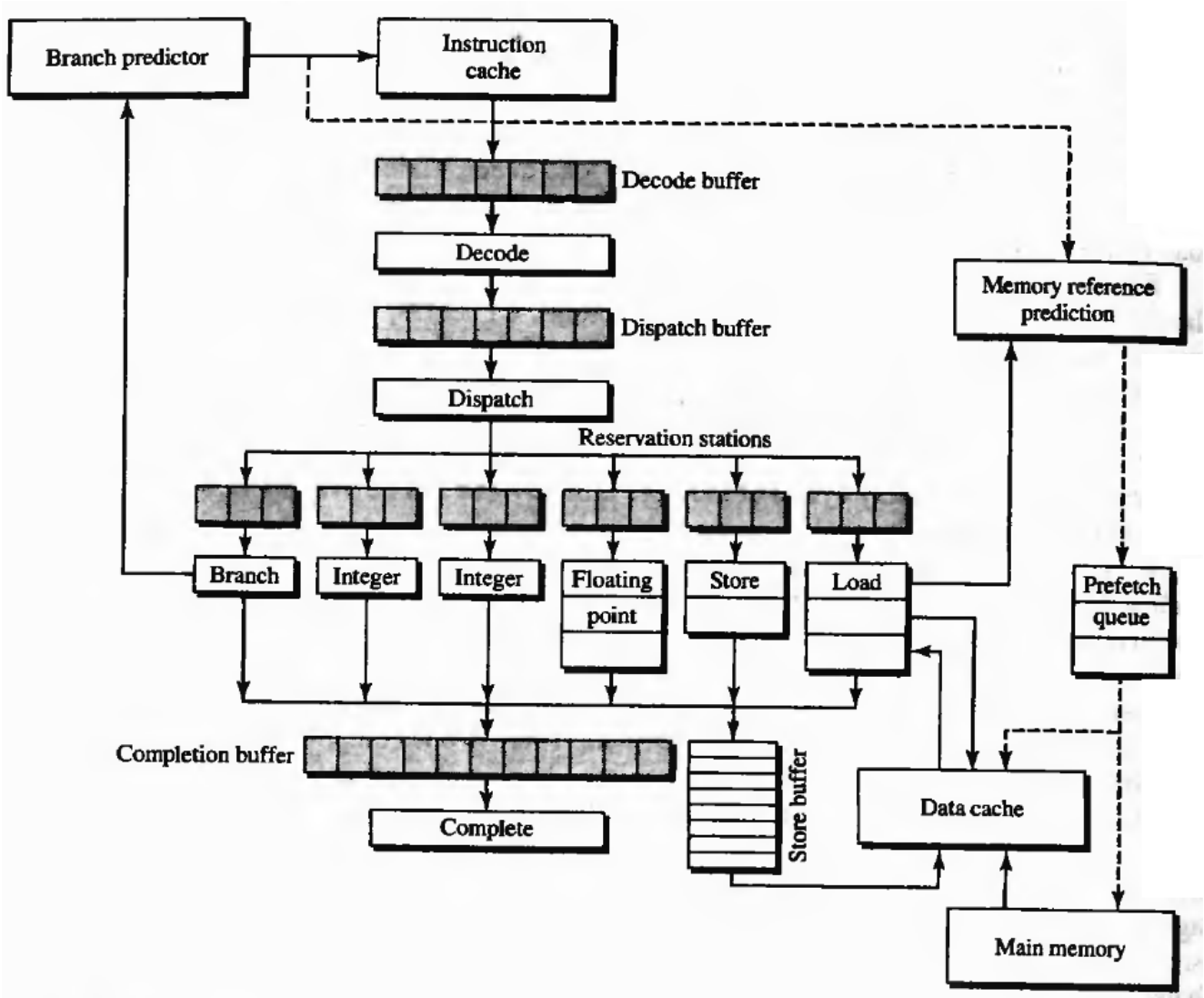
Other memory data flow techniques

Dualported and nonblocking data cache



Other memory data flow techniques

Prefetching data cache



What with reordering on multi-processor devices

Switch to terminal

Questions?

Sources:
Modern Processor Design - Funds of Superscalar Processors - J.Shen, M.Lipasti
http://preshing.com/20120515/memory-reordering-caught-in-the-act/