DLX Implementation

06-1

Set currently incomplete.

Material from chapter 3 of H&P.

EE 4720 Lecture Transparency. Formatted 10:36, 12 February 1999 from lsli06.

Unpipelined Implementation

06-2

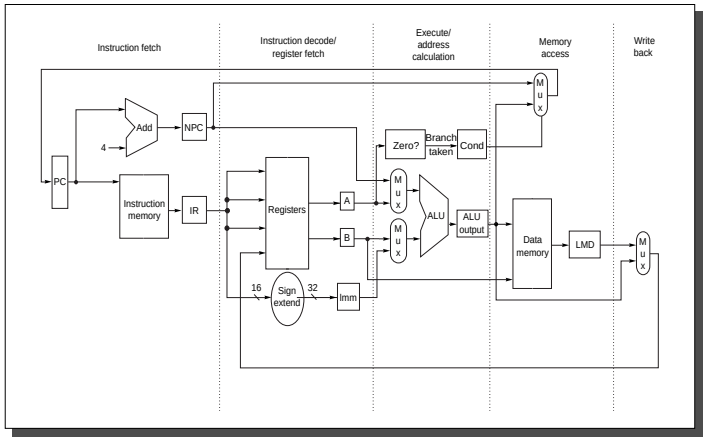
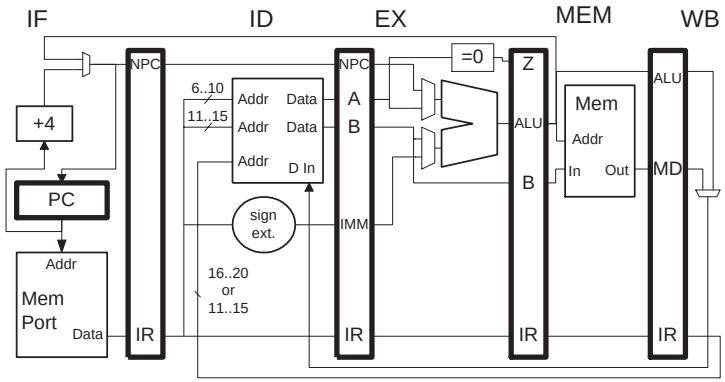


FIGURE 3.1 The implementation of the DLX datapath allows every instruction to be executed in four or five clock cycles.

EE 4720 Lecture Transparency. Formatted 10:36, 12 February 1999 from lsli06.

Pipelined Implementation

06-3



EE 4720 Lecture Transparency. Formatted 10:36, 12 February 1999 from lsli06.

Pipeline Details

06-4

Pipeline Segments

Divide pipeline into *segments*.

Each segment occupied by at most one instruction.

At any time, different segments can be occupied by different instructions.

Segments given names: IF, ID, EX, MEM, WB

Pipeline Registers

Registers written at end of each cycle.

To emphasize role, drawn as part of dividing bars.

Registers named using pair of segment names and register name.

For example, `IF/ID.IR`, `ID/EX.IR`, `ID/EX.A`.

EE 4720 Lecture Transparency. Formatted 10:36, 12 February 1999 from lsli06.

Pipeline Execution Diagram

06-5

Pipeline Execution Diagram

Used to show how instructions execute.

Along horizontal axis show time, vertical axis static instructions.

Label points with segment that instruction is in.

```
Cycle           0   1   2   3   4   5   6
add r1, r2, r3  IF  ID  EX  MEM WB
and r4, r5, r6      IF  ID  EX  MEM WB
lw  r7, 8(r9)           IF  ID  EX  MEM WB
```

A vertical slice (*e.g.*, at cycle 3) shows processor activity at that time.

In such a slice **a segment should appear at most once** . . .

. . . if it appears more than once execution not correct . . .

. . . since a segment can only execute one instruction at a time.

EE 4720 Lecture Transparency. Formatted 10:36, 12 February 1999 from lsli06.

Instruction Decoding and Pipeline Control

06-6

Pipeline Control

Setting control inputs to devices including . . .

. . . multiplexor inputs . . .

. . . function for ALU . . .

. . . operation for memory . . .

. . . whether to clock each register . . .

. . . *et cetera*.

Options for controlling pipeline:

- Magic Cloud
- Decode in ID
  Determine settings in ID, pass settings along in pipeline latches.
- Decode in Each Stage
  Pass opcode portions of instruction along.

  Decoding performed as needed.

EE 4720 Lecture Transparency. Formatted 10:36, 12 February 1999 from lsli06.

06-7

Real systems decode in ID.

For clarity, diagrams misleadingly imply decoding in stage needed . . .

. . . by passing entire instruction along.

EE 4720 Lecture Transparency. Formatted 10:36, 12 February 1999 from lsli06.

Pipeline Hazards

06-8

A *hazard* is a potential execution problem due to the hardware.

Hazards are caused by overlapping of instructions.

Hazards are avoided using *interlocks* or other means.

Hazard Types:

- *Structural Hazard*
  Needed resource not available.
- *Data Hazard*
  Needed value (written by previous instruction) not available.
- *Control Hazard*
  Needed instruction not available or wrong instruction executing.

EE 4720 Lecture Transparency. Formatted 10:36, 12 February 1999 from lsli06.

Data Hazards

Identified by acronym indicating correct operation.

- *RAW:* Read after write.
- *WAR:* Write after read.
- *WAW:* Write after write.

DLX implementation above only subject to RAW hazards.

RAR not a hazard since read order irrelevant (without an intervening write).

EE 4720 Lecture Transparency. Formatted 10:36, 12 February 1999 from lsli06.

Interlocks

When threatened by a hazards:

- *Stall* (Pause a part of the pipeline.)
  Stalling avoids overlap that would cause error.

  This does slow things down.
- Add hardware to avoid the hazards.
  Details of hardware depend on hazard and pipeline.

  Several will be covered.

EE 4720 Lecture Transparency. Formatted 10:36, 12 February 1999 from lsli06.

Structural Hazards

Cause: two instructions simultaneously need one resource.

Solutions:

Stall.

Duplicate resource.

Covered in more detail with floating-point instructions.

EE 4720 Lecture Transparency. Formatted 10:36, 12 February 1999 from lsli06.

Data Hazards

Chapter-3 DLX Subject to RAW Hazards.

Execution of code with two hazards, on `r1` and `r4`.

Hazard avoided by stalling.

```
add  r1, r2, r3  IF  ID  EX  MEM WB
sub  r4, r1, r5      IF  ID  --------->  EX  MEM WB
and  r6, r1, r8          IF  --------->  ID  EX  MEM WB
xor  r9, r4, r11                         IF  ID  ----->  EX  MEM WB
```


EE 4720 Lecture Transparency. Formatted 10:36, 12 February 1999 from lsli06.