**Date 11/14/2017**

**Step 2**

**Points looked at:**

1. **What is a pipeline?**

Wiki article : <https://en.wikipedia.org/wiki/Pipeline_(computing)> ,

Article 2 : <http://whatis.techtarget.com/definition/pipelining> ,

Summary: a pipeline is not **HARDWARE** it is the movement of instructions in a continues way to the processor which lets the ability to fetch next instruction and execute past instruction while holding them in a buffer next to the processor in the action called Pipelining and with put them the processor will have to first fetch the instructing then do the operation that it calls stalls until it ends then will fetch the next instruction and so on.

1. **What is Scoreboarding?**

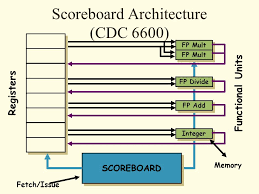
Wiki article : <https://en.wikipedia.org/wiki/Scoreboarding> ,

Article 2:

<https://www.cs.umd.edu/class/fall2001/cmsc411/projects/dynamic/scoreboard.html> this link has an example link in the bottom of the page helpful to look at.

Summary: socreboarding was first set for CDC (Control Data Corporation) 6600 supercomputers series, it is a **hardware** item that got functional unites (FU) that issue, execute instructions in an order that the scoreboards fits **(in order issue but out of order execution)** with its algorithm it takes care of different kind of hazards that could arise like (WAW (Write After Write), RAW (Read After Write) and WAR (Write After Read)) by stalling the instructions until the hazard is gone.

**Scoreboard structure**

****

FP: Floating point

1. **What is WAW (Write After Write), WAR (Write After Read) and RAW (Read After Write) and Structural hazards?**

Wiki article:

<https://en.wikipedia.org/wiki/Hazard_(computer_architecture)>

in the types section.

**WAW:** is write after write hazard which happens when to instructions want to right in the same register so the second instruction will have to stall until the first one finishes writing.

**WAR:** is write after read hazard which happens in the following case if we had two instructions and the first is reading from some register that the second instruction will right in it, in this case the second instruction will be stalled until the first instruction completes and reads it data.

**RAW:** is read after write hazard which happens in the following case if we had two instructions and the first is writing in a register which the second instruction needs data from so that will result in a stall on the second instruction until the first one completes and writes its data.

**Structural hazard:** happen in one simple way, when we have two same operation instruction and only one FU (Functional Unit) so the first instruction will busy up the FU then the second will have to stall until there is FU with the same operation free for it.

1. **What is a functional unit (FU)?**

Wiki article: <https://en.wikipedia.org/wiki/Execution_unit> ,

Summary: we can imagine it as unite which takes care of some kind of operation like an ADD functional unit which till take care of both add and sub operations and a MULT functional unit which will take care of multiplication operations.

1. **Scoreboard functionality or what are the stages that the scoredboard takes to handle an instruction?**

**Issue:** An instruction is issued if:

* The needed functional unit is free (there is no structural hazard)
* No functional unit has a destination operand equal to the destination of the instruction (resolves WAW hazards)

**Read:** Wait until no data hazards, then read operands:

* Performed in parallel for all functional units
* Resolves RAW hazards dynamically

**Execute(EX):** Normal execution

* Notify the scoreboard when ready

**Write:** The instruction can update destination if:

* All earlier instructions have read their operands (resolver WAR hazard)

1. **What is the data structure of the scoreboard?**

1. **Instruction status** – keeps track of which of the 4 steps the

instruction is in

2. **Functional unit status** – Indicates the state of the functional

unit (FU). 9 fields for each FU:

Busy: Indicates whether the unit is busy or not

Op: Operation to perform in the unit (e.g. add or sub)

Fi: Destination register name

Fj, Fk: Source register names

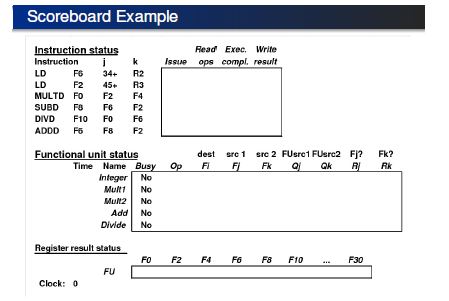
Qj, Qk: Name of functional unit producing regs Fj, Fk

Rj, Rk: Flags indicating when Fj and Fk are ready

3. **Register result status** – Indicates which functional unit will

write each register, if any

**Scoreboard example and how the registers and functional unit status look**

****

1. **Pros and cons of scoreboaring?**

**Pros:**

* managing multiple FUs
* out-of-order execution of multi-cycle operations
* maintaining all data dependencies (RAW, WAW, WAR)

**Cons:**

* single issue scheme, however: scheme is extendable to multiple-issue
* in-order issue
* anti-dependencies and output dependencies may lead to WAR and WAW stalls,
* no forwarding hardware =) all results go through the registers

**Helpful links:**

1. downloaded pdf important to read <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=6&cad=rja&uact=8&ved=0ahUKEwj6nIjlnsDXAhWBNRQKHel0DxcQFghBMAU&url=http%3A%2F%2Fwww.eit.lth.se%2Ffileadmin%2Feit%2Fcourses%2Feitf20%2FL4-handout4.pdf&usg=AOvVaw2QJbWuPktKQK_gS3v4cHNK>
2. another pdf with examples on scoreboarding and Tomasulo algorithm will see it in the next step <https://people.eecs.berkeley.edu/~pattrsn/252F96/Lecture04.pdf>