Not sure about “wraparound” signals

* Instruction (starts at IF)
  + ID -> Instruction\_ID (used in register file, control unit)
  + EX-> Instruction\_EX (used in alu, Alu control, mux\_2)
  + MEM -> Instruction\_MEM (used in branch\_unit)
  + Immediate extended?

* rdata\_ext (starts at instruction memory, IF)
  + not used further? -> output !
* Current\_Pc (starts at IF, program\_counter)
  + IF -> Current\_PC (used by instruction \_memory)
* Updated\_PC (starts at IF , program\_counter)
  + ID -> updated\_pc\_ID
  + EX ->updated\_pc\_EX
  + MEM-> updated\_pc\_MEM (used by branch unit)
* Regfile\_waddr (starts at EX, regfile\_dest\_mux)
  + MEM -> regfile\_waddr\_MEM
  + WB -> regfile\_waddr\_WB
  + IF -> regfile\_waddr\_IF
  + ID -> regfile\_waddr\_ID (used by register\_file)
    - Can be done using WB line only?
* Regfile\_wdata (starts at WB, regfile\_data\_mux)
  + IF -> regfile\_wdata\_IF
  + ID-> regfile\_wdata\_ID (used by register file)
* Next\_pc\_i ?????
* Regfile\_data\_1 (starts at ID, register\_file)
  + EX -> regfile\_data\_1\_EX (used by alu)
* Regfile\_data\_2 (starts at ID, register\_file
  + EX -> regfile\_data\_2\_EX (used by alu\_operand\_mux)
  + MEM -> regfile\_data\_2\_MEM (used by data\_memory)
* Alu\_operand\_2 (starts in EX, alu)
  + Only used by alu in EX
* Alu\_control (starts in EX, alu\_control)
  + Only used in EX
* Immediate\_extended (starts in IF, hard assign from instruction)
  + ID -> Immediate\_extended\_ID
  + EX -> Immediate\_extended\_EX (used by alu\_operand\_mux)
  + MEM -> immediate\_extended\_MEM (used by branch\_unit)
* Alu\_src (starts in ID, control unit)
  + EX-> alu\_src\_EX (used by alu\_operand\_mux)
* Alu\_op (starts in ID, control unit)
  + EX -> Alu\_op\_ex (used by alu\_control
* Reg dst (starts in ID, control unit)
  + EX -> reg\_dst\_EX (used by regfile\_dest\_mux)
* Alu\_out (starts in EX, alu)
  + MEM -> alu\_out\_MEM ( used by data memory)
  + WB -> Alu\_out\_wb (used by regfile\_data\_mux)
* Zero\_flag (starts in EX, Alu)
  + MEM -> zero\_flag\_MEM
  + WB -> Zero\_flag\_WB
  + IF-> Zero\_flag\_ID (used by program counter)
* Overflow ????
* Memwrite (starts in ID, control code)
  + EX -> mem\_write\_EX
  + MEM -> mem\_write\_MEM (used by data\_memory)
* Mem\_read (starts in ID, control unit)
  + EX -> mem\_read\_EX
  + MEM -> mem\_read\_MEM (used by data\_memory)
* Addr\_ext\_2 (starts at IF, input)
  + ID -> addr\_ext\_2\_ID
  + EX -> addr\_ext\_2\_EX
  + MEM -> addr\_ext\_2\_MEM (used by data\_memory)
* Wen\_ext\_2 (starts at IF, input)
  + ID-> wen\_ext\_2\_ID
  + EX -> wen\_ext\_2\_EX
  + MEM -> wen\_ext\_2\_MEM (used by data\_memory)
* Ren\_ext\_2 (starts at IF, input)
  + ID -> ren\_ext\_2\_ID
  + EX -> ren\_ext\_2\_EX
  + MEM -> ren\_ext\_2\_MEM (used by data\_memory)
* Wdata\_ext\_2 (starts at IF, input)
  + ID -> wdata\_ext\_2\_ID
  + EX -> wdata\_ext\_2\_EX
  + MEM -> wdata\_ext\_2\_MEM (used by data\_memory
* Dram\_data (starts in MEM, data\_memory)
  + WB -> Dram\_data\_WB (used by regfile\_data\_mux)
* Rdata\_ext\_2 (starts at MEM, data\_memory)
  + WB -> Rdata\_ext\_2 (used as output)
* Branch\_pc (starts in MEM, branch\_unit)
  + WB -> branch\_pc\_WB
  + IF -> branch\_pc\_IF (used by program counter)
* Jump\_pc(starts in MEM, branch\_unit)
  + WB -> jump\_pc\_WB
  + IF -> Jump\_pc\_IF (used by program counter)
* Addr\_ext (starts in IF, input)
  + Only used in IF
* Wen\_ext (starts in IF, input)
  + Only used in IF
* Ren\_ext (starts in IF, input)
  + Only used in IF
* Wdata\_ext (starts in IF, input
  + Only used in IF
* Branch (starts in ID, control unit)
  + EX -> branch\_EX
  + MEM -> branch\_MEM
  + WB -> Branch\_WB
  + IF -> Branch\_IF (used by program counter)
* Jump (starts in ID, control unit)
  + EX -> jump\_EX
  + MEM -> jump \_MEM
  + WB -> jump \_WB
  + IF -> jump \_IF (used by program counter)
* Mem\_2\_reg (starts in ID, control\_unit)
  + EX -> mem\_2\_reg\_EX
  + MEM -> mem\_2\_reg\_MEM
  + WB-> mem\_2\_reg\_WB (used by regfile\_data\_mux)
* Reg\_write (starts in ID, control unit)
  + Used in ID