6.823 Computer System Architecture
HAL 180 ISA and 6-Stage Pipelined Implementation

http://csg.csail.mit.edu/6.823/

Inspired by how the IBM 360 uses condition codes, Ben Bitdiddle designs the HAL 180 architecture, which features two flag registers. Table 1 describes these flags.

Name	Description
Sign Flag (SF)	Stores 1 if the result of the *last arithmetic or comparison instruction* was negative, 0 if it was positive
Zero Flag (ZF)	Stores 1 if the result of the *last arithmetic, logical, or comparison instruction* was zero, and 0 if it was non-zero

Table 1. HAL 180 status flags.

Table 2 summarizes the different instruction types and the flags they read or write. The SF and ZF columns have an "R" when the instruction reads the status flag, a "W" if it writes the flag (and does not read it), or a blank if the instruction does not affect the status flag. For example, `JL` (jump if less than) reads SF; `ADD` writes all flags; and `JMP` (unconditional jump) does not affect any flag. Some instructions, like `CMP`, write the status flags but do not return any result.

Instruction	Description	SF	ZF
Arithmetic Instructions			
ADD *s1, s2*	$s1 \leftarrow s1 + s2$	W	W
SUB *s1, s2*	$s1 \leftarrow s1 - s2$	W	W
MUL *s1, s2*	$s1 \leftarrow s1 \times s2$	W	W
Logical Instructions			
AND *s1, s2*	$s1 \leftarrow s1 \,\&\, s2$		W
OR *s1, s2*	$s1 \leftarrow s1 \mid s2$		W
XOR *s1, s2*	$s1 \leftarrow s1 \wedge s2$		W
Comparison Instructions			
CMP *s1, s2*	$temp \leftarrow s1 - s2$	W	W
Jump Instructions			
JMP *target*	jump to the address specified by *target*		
JL *target*	jump to *target* if SF == 1	R	
JG *target*	jump to *target* if SF == 0 and ZF == 0	R	R
Memory Instructions			
LD *s1, s2*	$s1 \leftarrow M[s2]$		
ST *s1, s2*	$M[s1] \leftarrow s2$		

Table 2. HAL 180 instruction set.

Ben also designs a 6-stage pipelined implementation of the HAL 180. In this pipeline, the ALU takes three pipeline stages (E1, E2, and E3), and status flags are updated in stage E3. Table 3 describes each stage, and Figure 1 shows the datapath of this 6-stage pipelined architecture, highlighting the differences with a conventional MIPS pipeline.

Note that this implementation does not have any data bypass paths.

Stage	Description
Fetch and Decode Stage (FD)	Fetch an instruction from the instruction memory, decode the instruction, and fetch the register values from the register file. The status flag checking for conditional jumps is also done in this stage.
Execute Stage 1 (E1)	The first stage of the execution phase. Generate partial results and store them in the pipeline registers.
Execute Stage 2 (E2)	The second stage of the execution phase. Generate partial results and store them in the pipeline registers.
Execute Stage 3 (E3)	The final stage of the execution phase. Final results are generated and flag registers get updated if necessary.
Memory Stage (M)	Perform load/store from/to the data memory if necessary.
Writeback Stage (WB)	Write to the register file if necessary.

Table 3. HAL 180 pipeline stages.

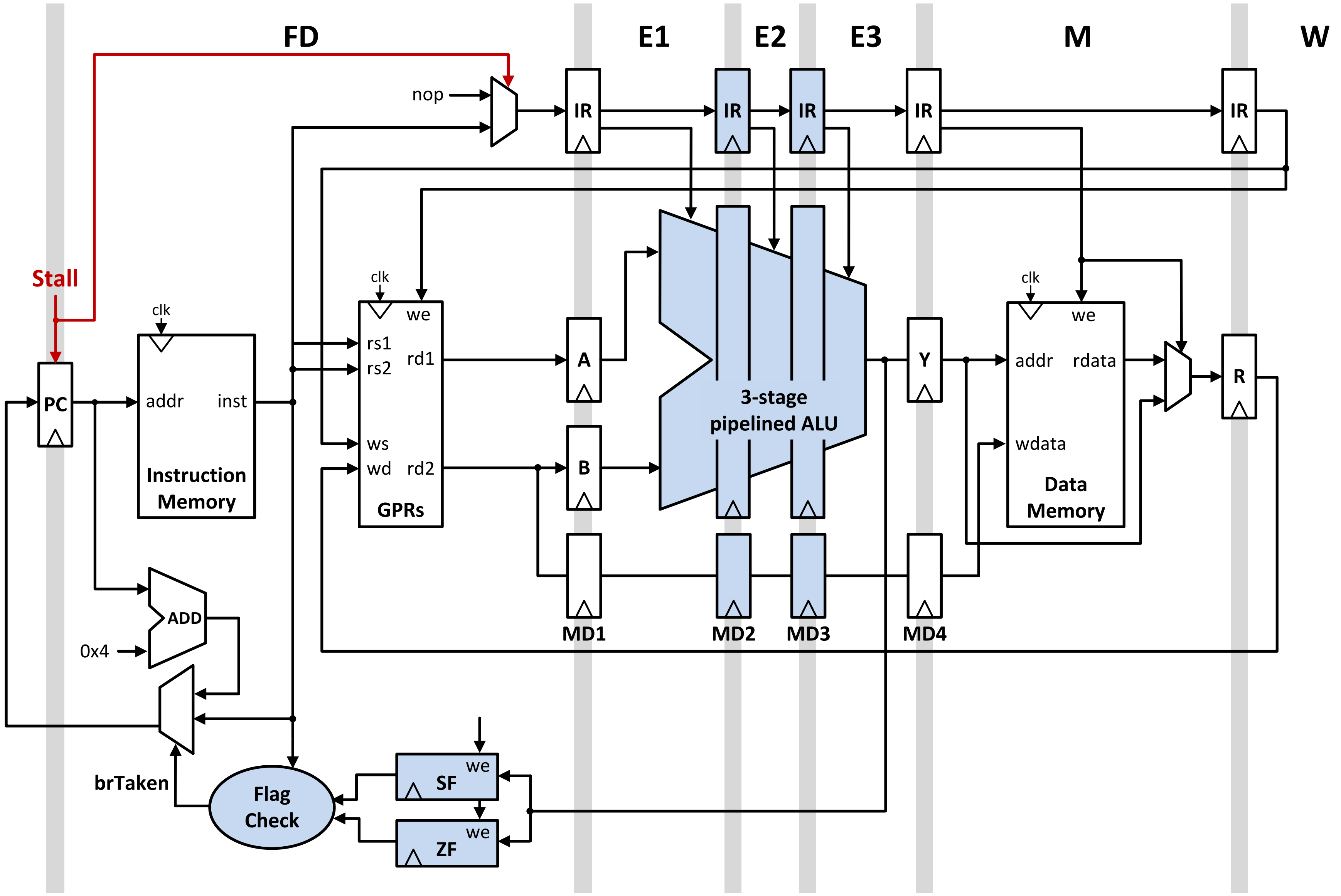

Figure 1. HAL 180 6-Stage pipelined implementation.