

Talos ES™

Programmers Guide

Copyright © 2023 by Bernardo Kastrup

All rights reserved

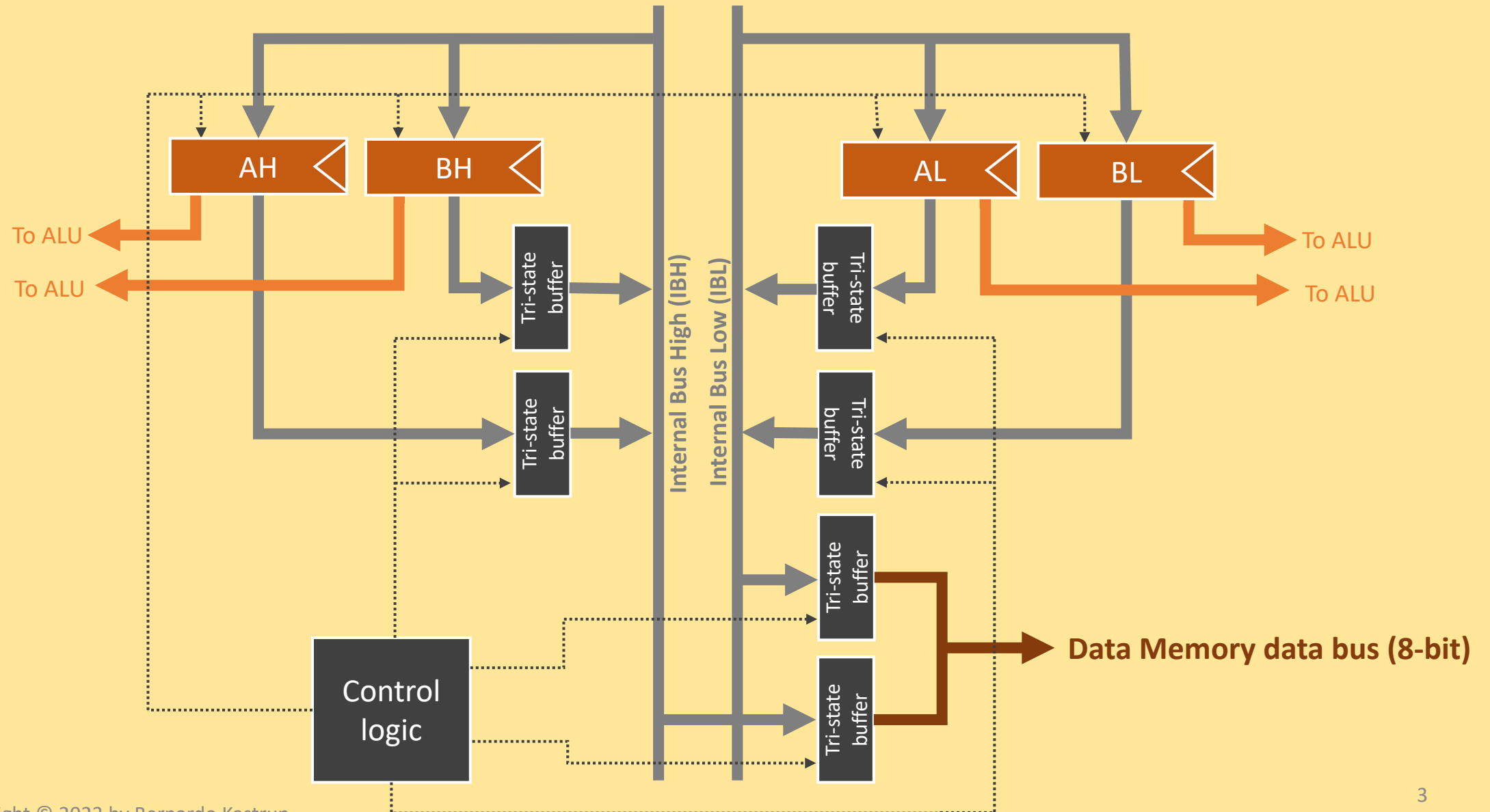
Provided as-is, without guarantees or
representations of any kind

Last edited: 11/19/23 1:48:54 AM

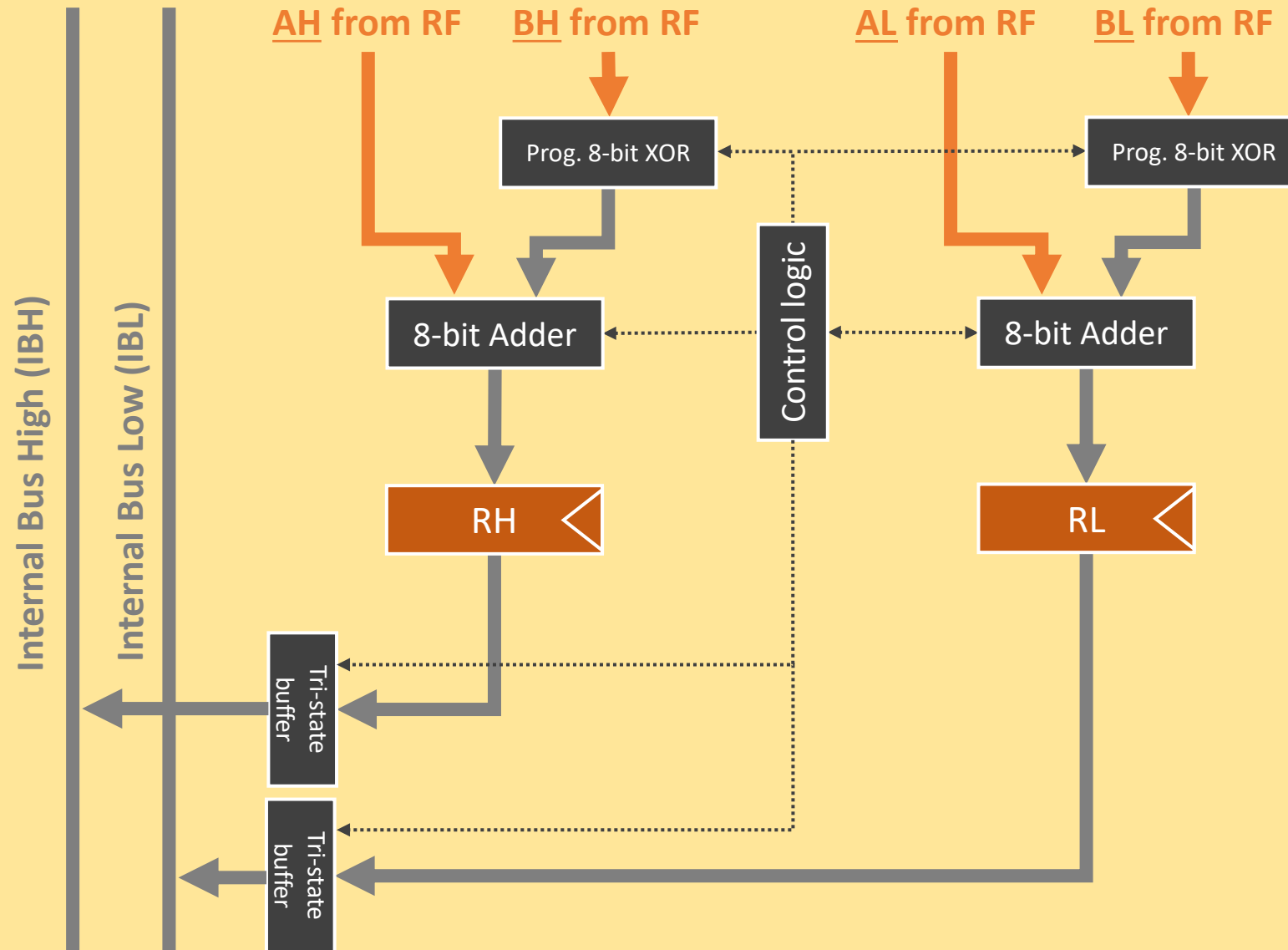
Talos ES™: Introduction

- Talos ES™ has a custom RISC CPU with a shallow, 3-stage pipeline
 1. Instruction fetch
 2. Instruction decode
 3. Instruction execution
- All operations are orthogonal:
 - Memory access operations
 - Register-to-register operations
 - ALU/control flow operations
- All pipeline stages take one clock cycle to complete
- There is just *ONE* delay slot after control flow (i.e. jump) operations
- The CPU can execute up to five operations in parallel, every cycle
- The CPU has a Harvard architecture, with separate data and instruction memories, each with its own address space
- The CPU has two 8-bit internal buses: Internal Bus High (IBH) and Internal Bus Low (IBL), through which register-to-register operations take place
- The CPU can do both 8-bit and 16-bit operations
 - During 8-bit operations, the two internal buses operate separately and in parallel
 - During 16-bit operations, the two internal buses are driven together

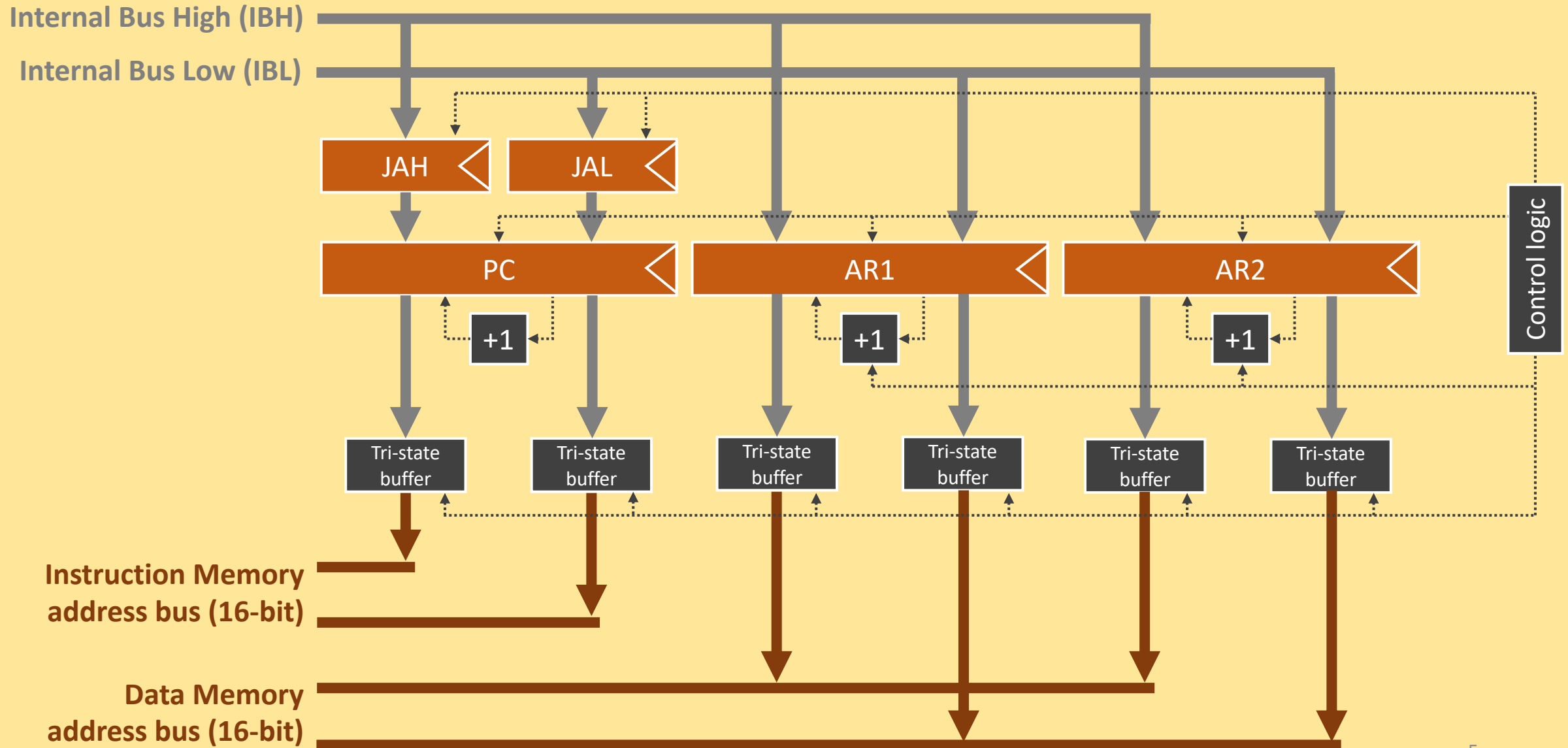
Talos CPU™, Register File (RF) overview



Talos CPU™, ALU overview



Talos CPU™, Addressing Logic (AL) overview



Talos ES™: Registers commentary (1/2)

- The CPU has four 8-bit general-purpose registers
 - **AH, BH, AL, BL**
- These four registers can be addressed as two general-purpose 16-bit registers
 - **A** is **AH,AL** and **B** is **BH,BL**
- All general-purpose registers can be both sources and destinations of register-to-register operations
- The results of ALU operations are stored in special-purpose registers **RH** and **RL**, which can be addressed together as 16-bit register **R**
- **RH** and **RL** can only be sources in register-to-register operations
 - **AH = RH** is OK, but **RH = AH** isn't
- Register-to-register operations cannot go across the internal buses
 - **BH = RH** works, but not **BH = RL**
 - **AH = BH** works, but not **AH = BL**
- There are four other special-purpose 16-bit registers
 - Program Counter (**PC**): contains the instruction memory address
 - Jump Address (**JA**) register: in a jump, the contents of this register are parallel-loaded into the **PC**
 - Address Registers 1 (**AR1**) and 2 (**AR2**): Drive the address bus of the external data memory

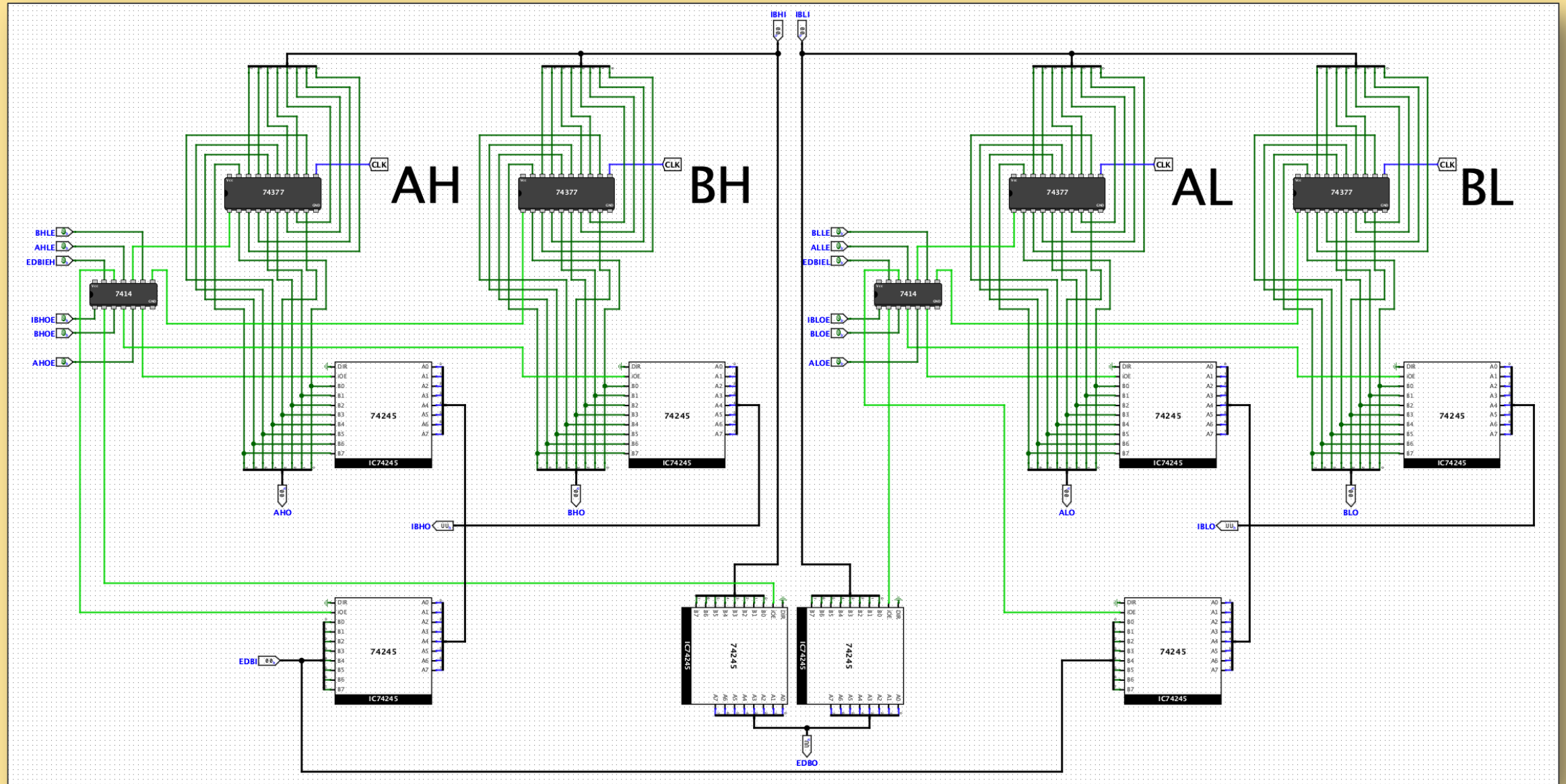
Talos ES™: Register commentary (2/2)

- The **PC** can either count (+1) or synchronously receive (via a parallel load) the contents of **JA**
- Unlike all other registers, the **PC** latches a new value on the falling edge of the clock
 - Talos ES™ has *one* delay slot
- **JA** can be addressed as two 8-bit registers: **JAH** and **JAL**
- **JAH** and **JAL** are never source registers in register-to-register operations (they only drive the **PC**), but always destinations
 - **JAH** = **BH** is OK
 - But **BH** = **JAH** does not work
- Similarly, **AR1** and **AR2** are never source registers for register-to-register operations; they only drive the external data memory's address bus
 - **AR1** = **B** is OK
 - But **B** = **AR1** does not work
- **AR1** and **AR2** must *always* be addressed as 16-bit registers
- **AR1** and **AR2** have built-in counters, and can have their contents incremented by 1 (at a time) without ALU intervention
 - In assembly: **AR1++** and **AR2++**

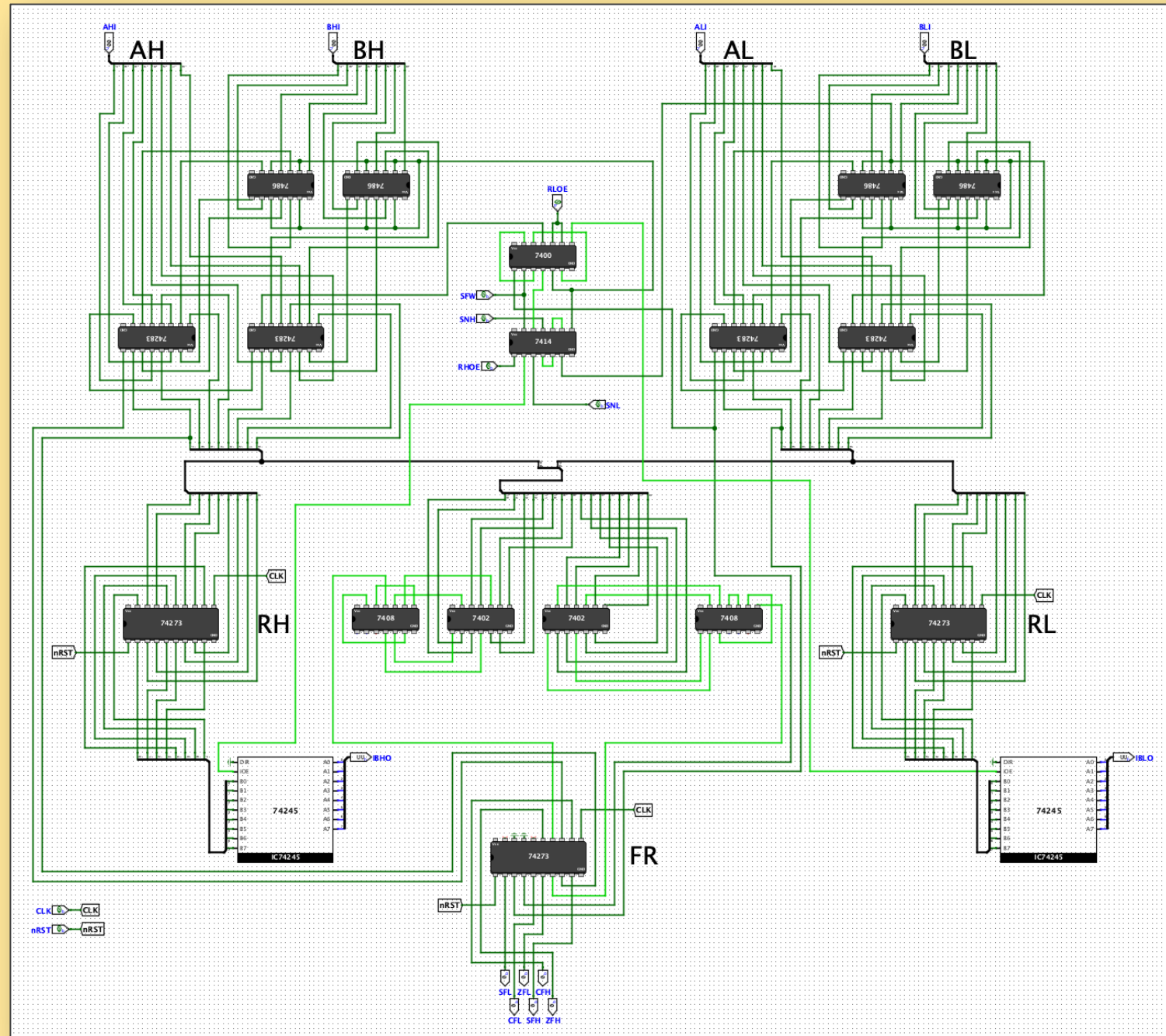
Talos ES™: ALU commentary

- Unless otherwise specified in assembly, the high side of the ALU always performs an addition (**RH = AH + BH**), while the low side always performs a subtraction (**RL = AL – BL**)
 - These operations do *not* need to be specified in assembly, they occur in parallel with whatever is specified in assembly, and complete one cycle after the operands are loaded
- In subtractions, the subtrahend must be in **BH** or **BL**, while the minuend must be in **AH** or **AL**
 - **RH=AH-BH** and **RL=AL-BL** are OK
 - But **RH=BH-AH** and **RL=BL-AL** will not work

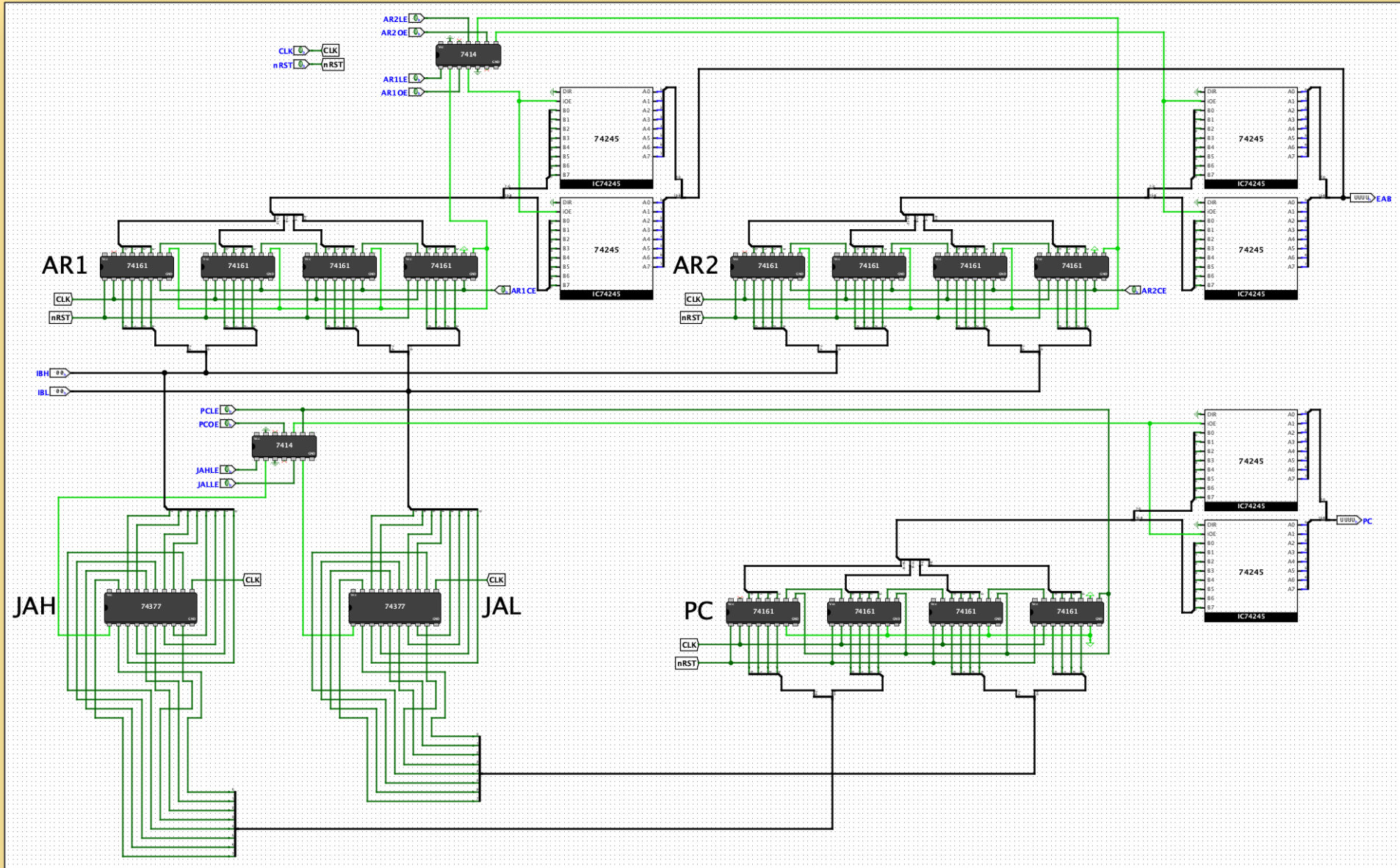
RF circuit



ALU circuit



AL circuit



CTRL circuit

