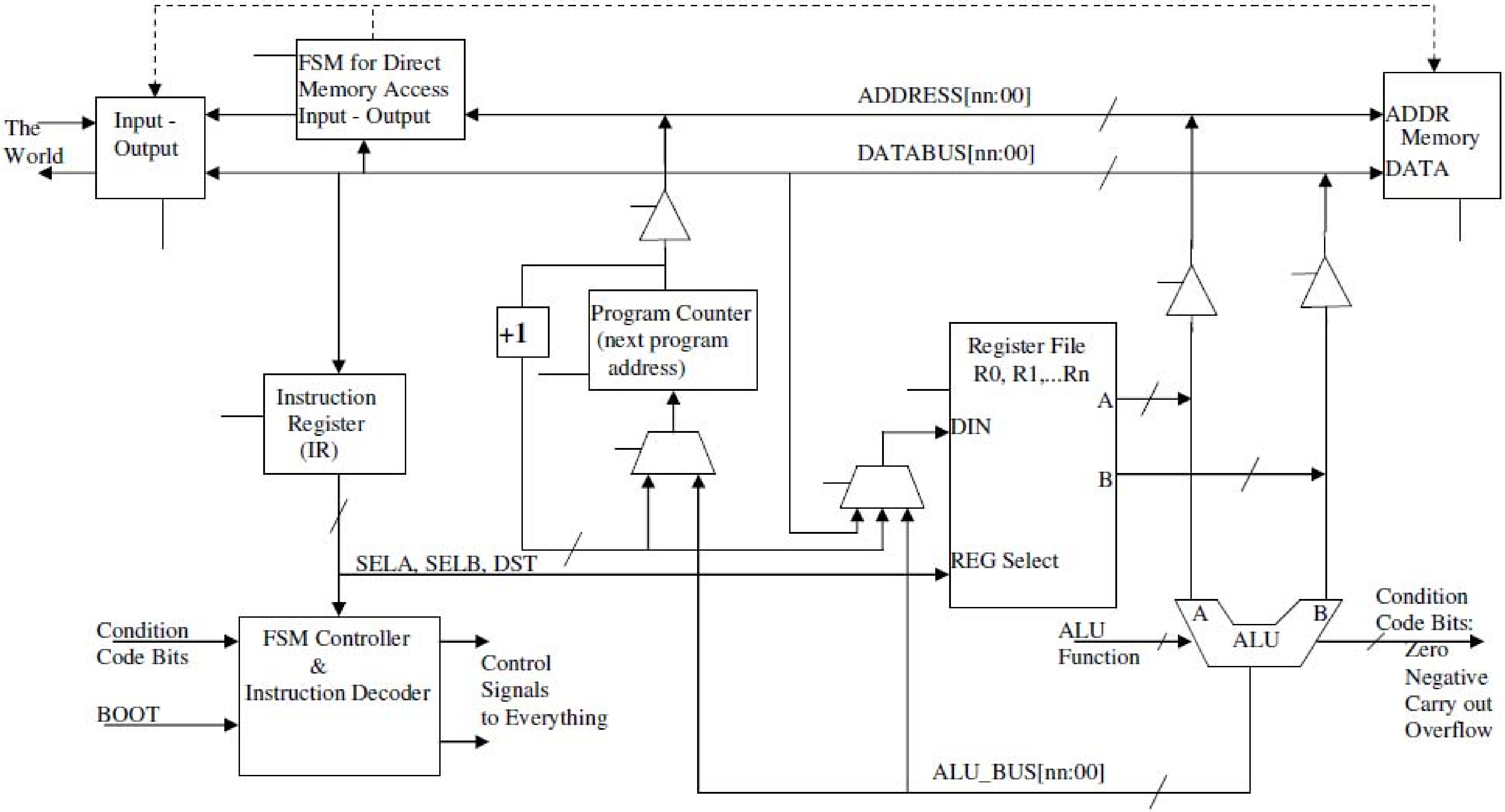
The Idea of a Register Transfer Machine

- A Template for the Strategic Deployment of Registers, Buses, Logic, and Memory
- May be Fixed or Programmable Hardware
  - Fixed Data Processing (DSP) often implemented in FPGAs
  - Stored program control machine ⇔ Computer
  - FPGAs can implement computer but dedicated processors usually more efficient
  - Hybrid Reconfigurable systems are possible
- An Example
  - Abstracted and Simplified Version of Lab D

This Example RTM Has:

- Resources:
  - Memory
  - Arithmetic Logic
  - Register File
  - Special Purpose Registers
  - I/O – Input/Output Path
- Buses:
  - Data Bus – goes everywhere
  - Address Bus – determines source and destination for memory or I/O data
- Control Signals:
  - Derived separately from data captured in Instruction Register (IR)
  - Originate in FSM Instruction Decoder



FSM for Direct Memory Access Input - Output
The World
Input - Output
ADDRESS[nn:00]
DATABUS[nn:00]
ADDR
Memory
DATA
+1
Program Counter (next program address)
Register File R0, R1,...Rn
A
B
DIN
REG Select
Instruction Register (IR)
SELA, SELB, DST
Condition Code Bits
BOOT
FSM Controller & Instruction Decoder
Control Signals to Everything
ALU Function
A
ALU
B
Condition Code Bits:
Zero
Negative
Carry out
Overflow
ALU_BUS[nn:00]

Notes to Block Diagram

Simple RTM System with Program Control - aka "*A Computer*": This is most of Lab D and its design is my variations on a theme by ARM. The short lines on the sides of almost all the functional blocks are the control lines for that block. For example, the short line just above the DIN bus on the left side of the register file determines whether the register file is written on the current clock cycle. The control lines all originate in the Instruction Decoder. There are a number of ways to implement the controller block, some of which can use FSMs to produce sequences of control signals. The slash lines across many of the wires represent buses of whatever the standard data bus width is for that machine. In Lab D this is 16 bits. The use of tristate drivers for bus multiplexing is schematic of function rather than implementation. Some or all of that function may be done with multiplexers instead. The BOOT line at the lower left is asserted at power turn-on and forces the controller to fetch an instruction from a particular location in memory, often at address 0x00000.

What Does It Do?

- Some data words are instructions
- Typical instruction: ADD R07, R02, R03
  - If the binary word for this instruction is in IR, then slightly later the contents of R07 are replaced by the sum of the contents of R02 and R03
- Controller sets control signals:
  - To read R02&R03, activate addition, write answer to R02 – for fast ALU/registers only one cycle to execute
  - Also gets next instruction by increment PC, put PC onto address bus, enable IR in same cycle as execution

The “Take-Aways”

- RTM style and strategy common to the design of most complex systems. Think movement of data!
- Primary features: buses, resources (memory, arithmetic, fast register file(s), PC and IR registers, etc.) and FSM(s). Hence the term RTM machine.
- Blocks are connected together with buses and data is controlled by “instructions” from the data stream.
- Read Lab D for a more detailed discussion.