CSC 220: Computer Organization

Unit 11
Datapath Design

Prepared by:

Md Saiful Islam, PhD

Department of Computer Science

College of Computer and Information Sciences

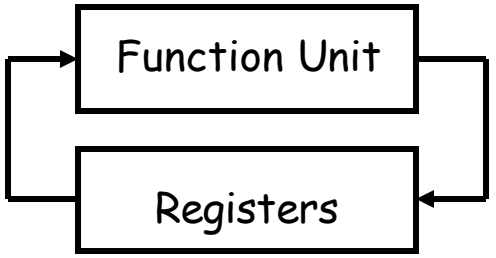
Overview

- Guiding principles for datapath design
- Register file
- Datapath Representation
- Accessing RAM
- The Control Word

Chapter-8

M. Morris Mano, Charles R. Kime and Tom Martin, **Logic and Computer Design Fundamentals**, Global (5th) Edition, Pearson Education Limited, 2016. ISBN: 9781292096124

Datapath Design

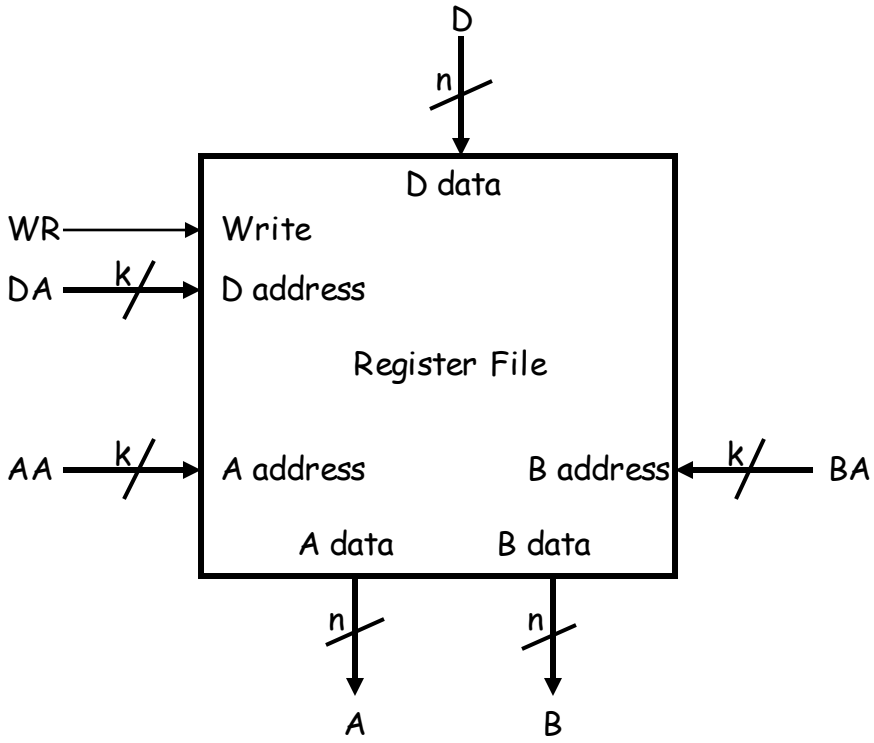


Guiding principles for basic datapath:

- A limited set of registers serve as fast temporary storage
  - A set of registers with common access resources called a ***register file***
- Microoperation implementation
  - ***Function Unit (ALU)*** - shared resource for implementing arithmetic, logic, and shift microoperations
  - **Buses** - shared transfer paths
  - For given microoperation, we need to specify;
    - Source registers
    - Destination register
    - Operation to perform
- A larger, but slower, random-access memory is also available.
- The datapath completes a single microoperation each clock cycle.

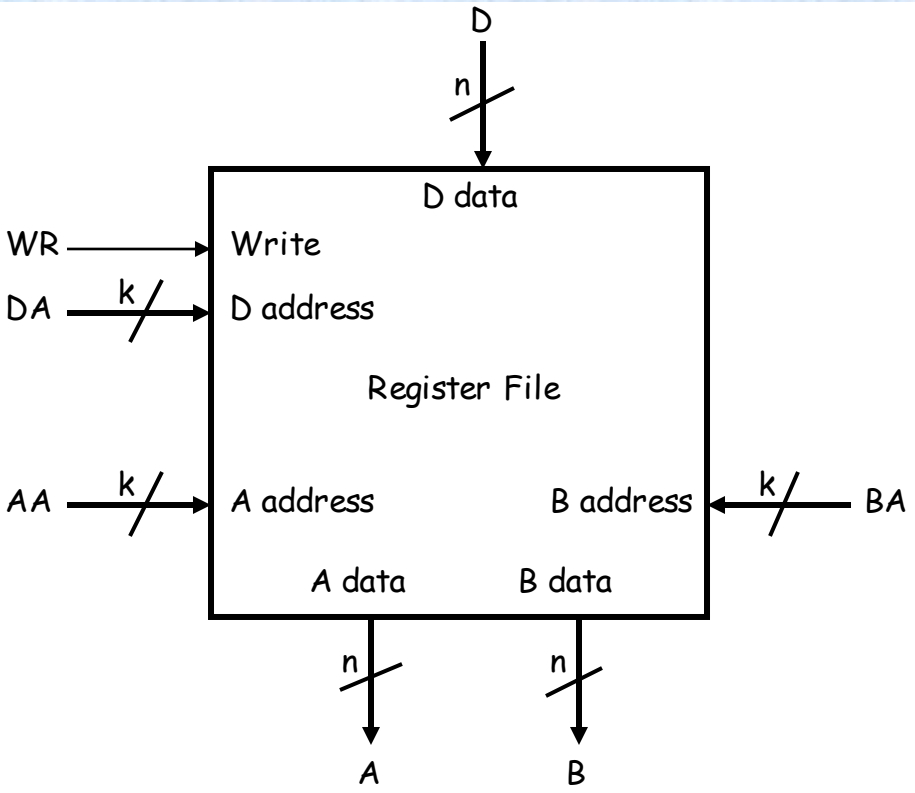
Register file

- **Register file:** A register file is an array of fast registers
- Modern processors contain a number of registers grouped together in a register file.
- The register file appears like a memory based on clocked flip-flops (the clock is not shown)
- Much like words stored in a RAM, individual registers are identified by an address.
- Here is a block symbol for a 2^k x n register file.
  - There are 2^k registers, so register addresses are k bits long.
  - Each register holds an n-bit word, so the data inputs and outputs are n bits wide.



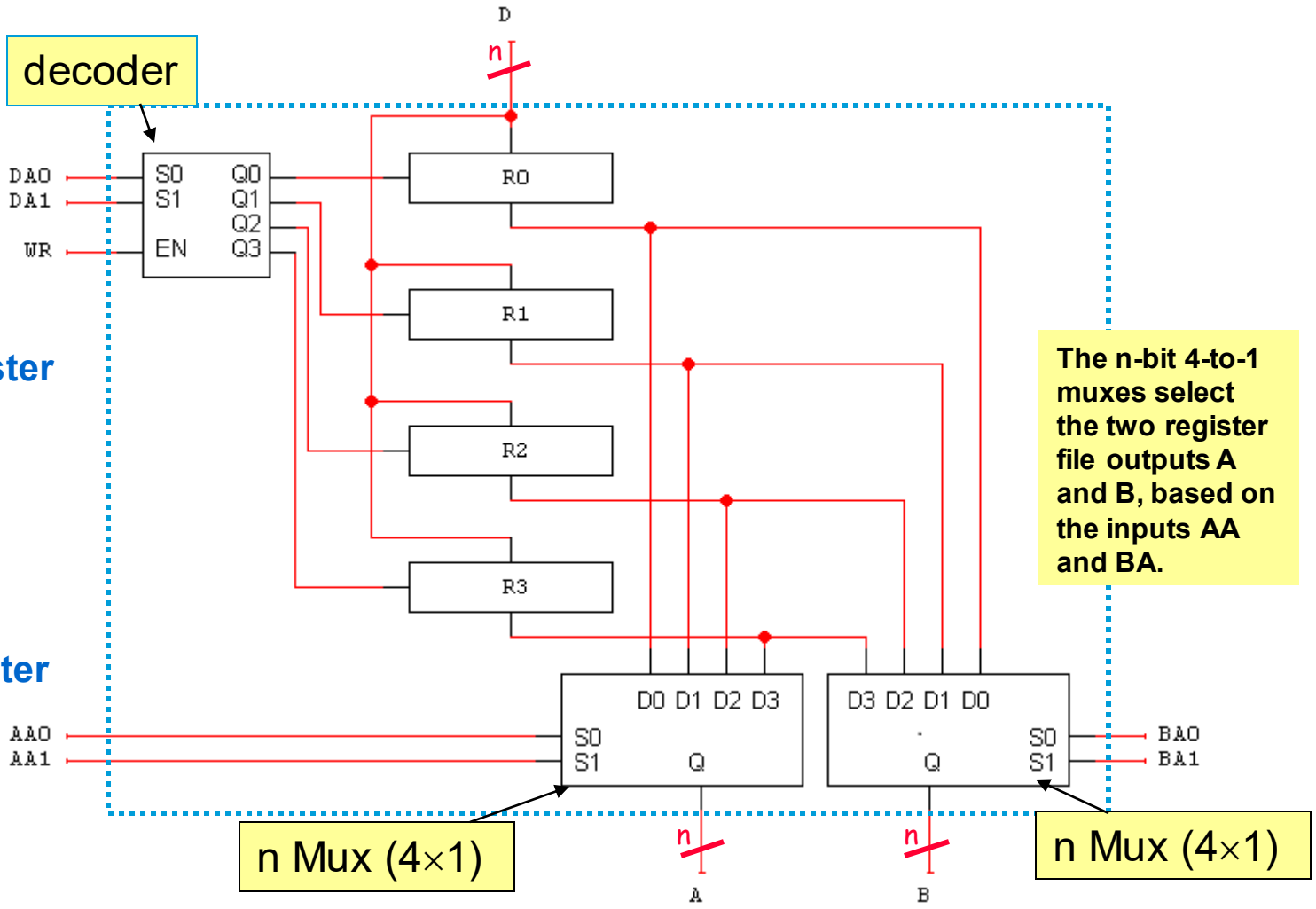
Accessing the register file …

- You can read *two* registers at once by supplying the AA and BA inputs. The data appears on the A and B outputs.
- You can write to a register by using the DA and D inputs, and setting WR = 1.
- These are registers so there must be a clock signal, even though we usually don't show it in diagrams.
  - We can read from the register file at any time.
  - Data is written only on the positive edge of the clock.



What's inside the register file …

- A 4 x n register file (2^k x n = 4 x n)

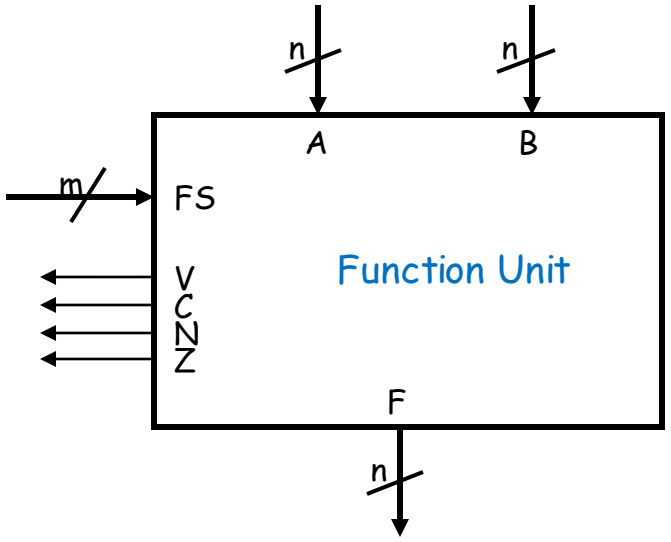


The Function Unit

Function Unit = ALU + Shifter

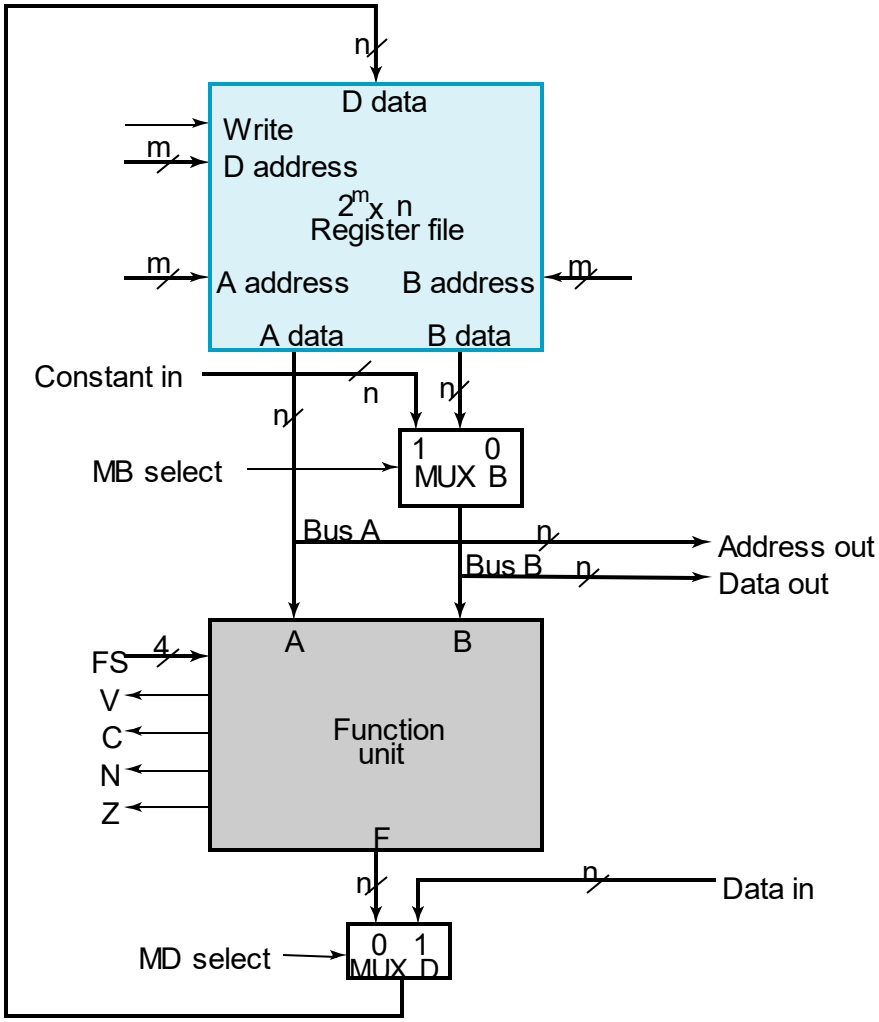
- The function select code FS is 4 bits long,
- There are only 15 different functions here.

FS	Operation
0000	F = A
0001	F = A + 1
0010	F = A + B
0011	F = A + B + 1
0100	F = A + B'
0101	F = A + B' + 1
0110	F = A - 1
0111	F = A
1000	F = A ∧ B (AND)
1001	F = A ∨ B (OR)
1010	F = A ⊕ B (XOR)
1011	F = A'
1100	F = B
1101	F = sr B (shift right)
1110	F = sl B (shift left)



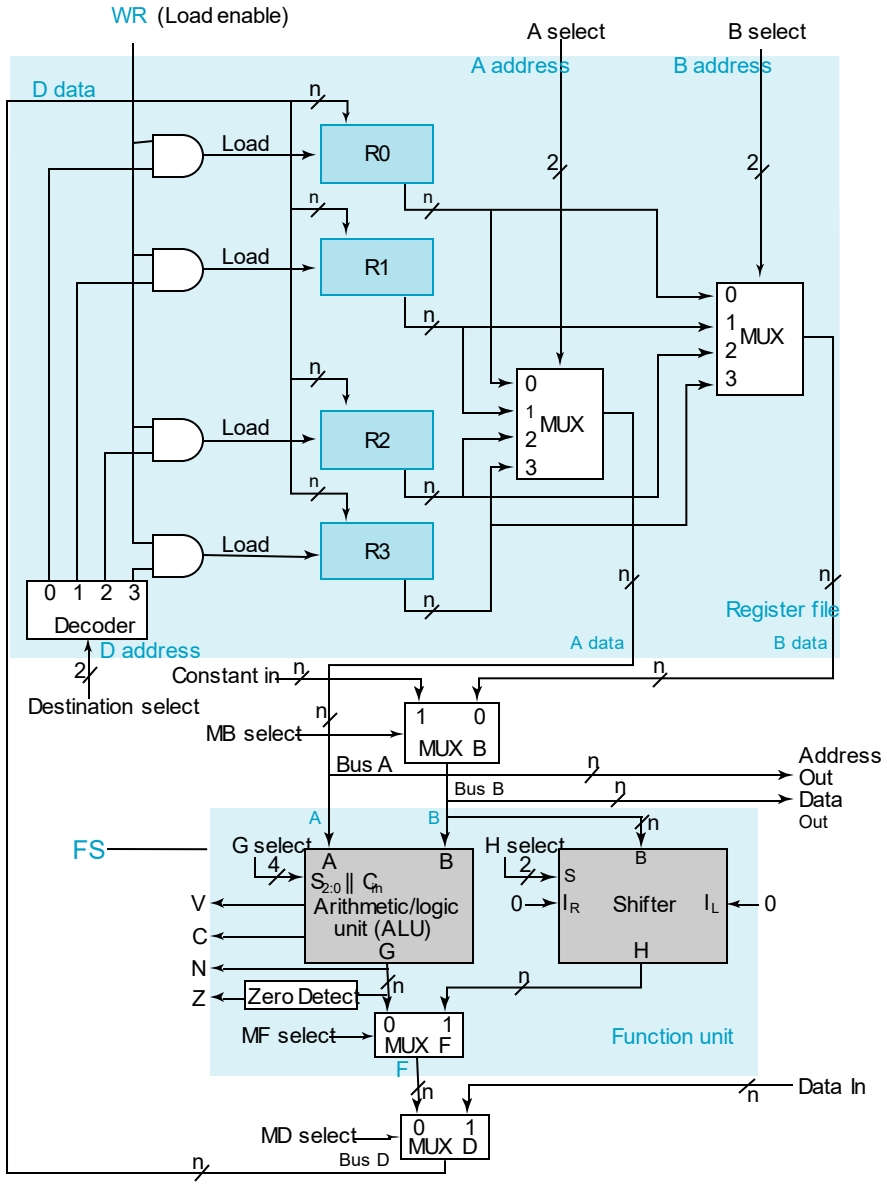
Datapath Representation

- **Datapath = Function unit + Register file:**
- Muxes (B, D) and buses handle data transfers



Datapath Example …

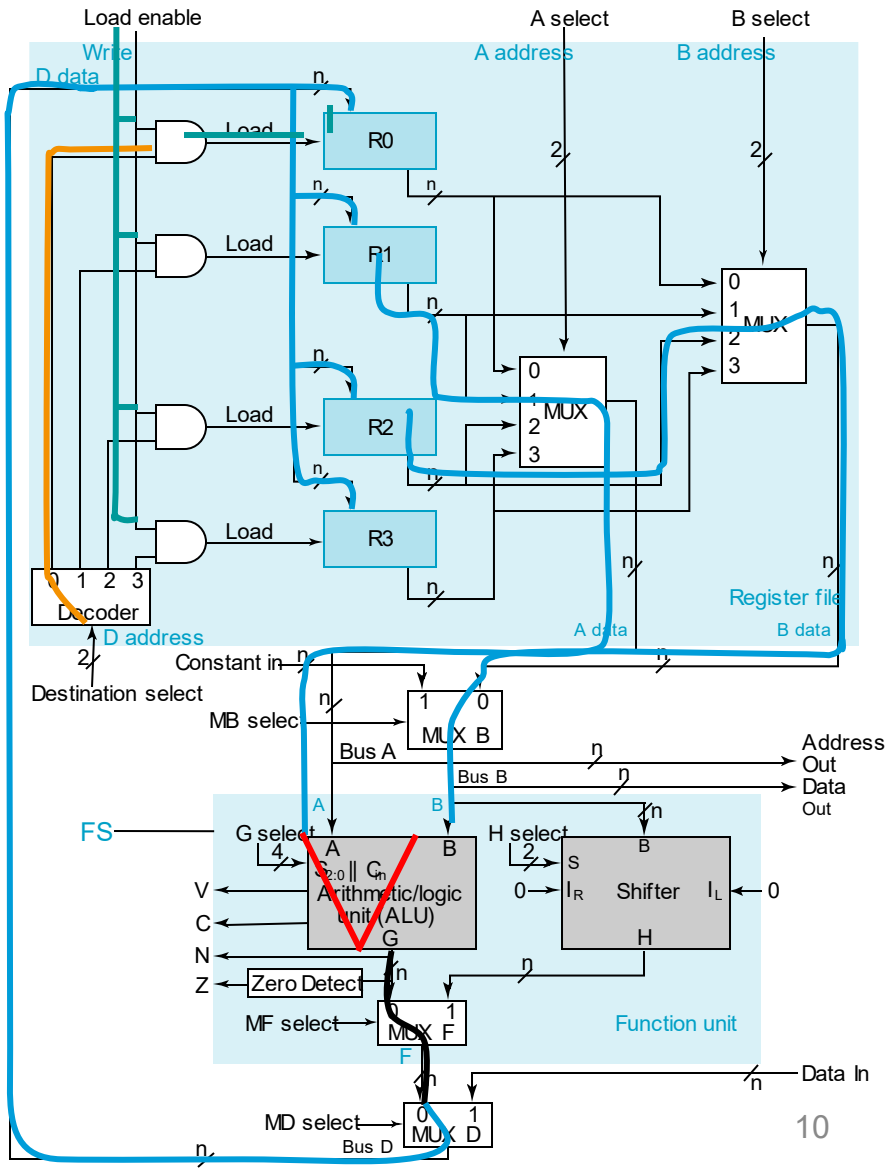
- Four parallel-load registers (R0-R3)
- Two mux-based register selectors
- Register destination decoder
- Mux B for external constant input
- Buses A and B with external address and data outputs
- Function Unit: ALU and Shifter
- Mux D for external data input



Performing a Microoperation …

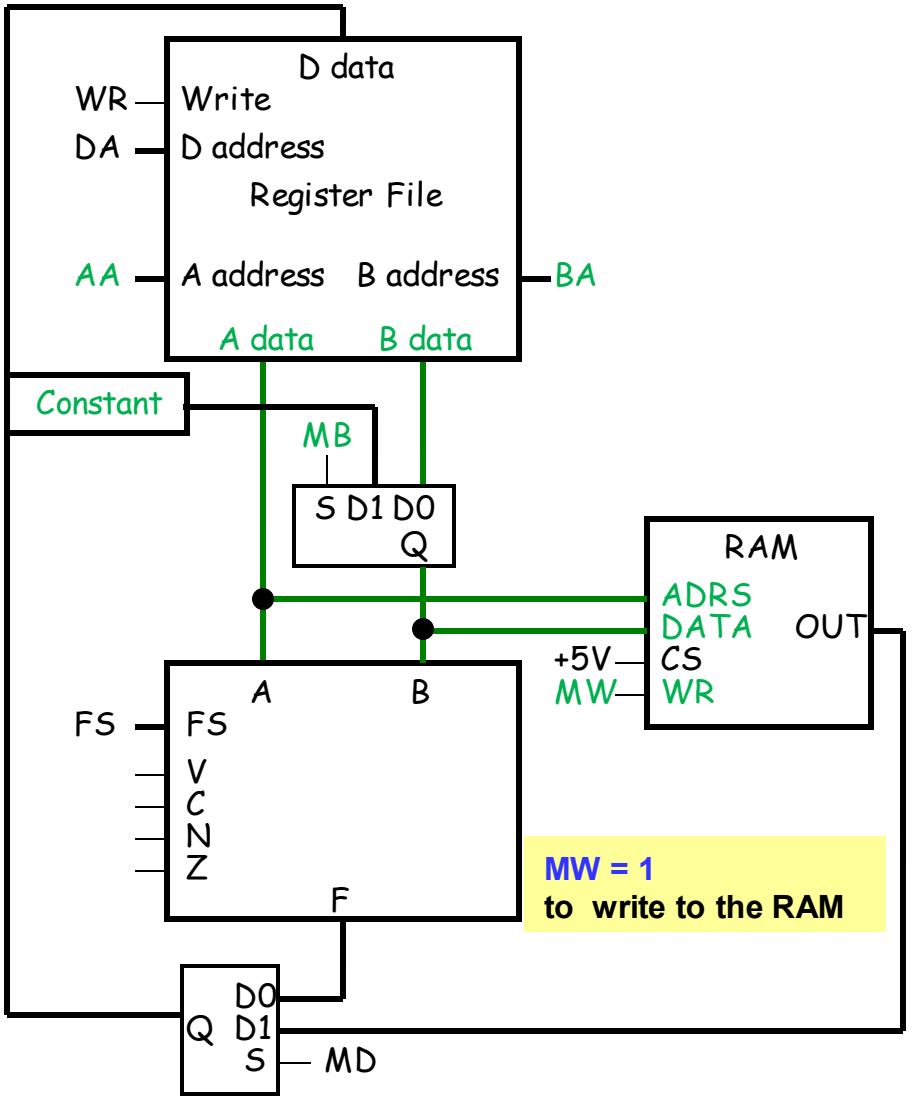
Microoperation: R0 ← R1 + R2

- Apply 01 to A select to place contents of R1 onto Bus A
- Apply 10 to B select to place contents of R2 onto B data
- Apply 0 to MB select to place B data on Bus B
- Apply 0010 to FS select to perform addition G = Bus A + Bus B
  - Apply 0 to MD select to place the value of G onto BUS D
  - Apply 00 to Destination select to enable the Load input to R0
- Apply 1 to Load Enable to force the Load input to R0 to 1 so that R0 is loaded on the clock pulse (not shown)
- The overall microoperation requires 1 clock cycle



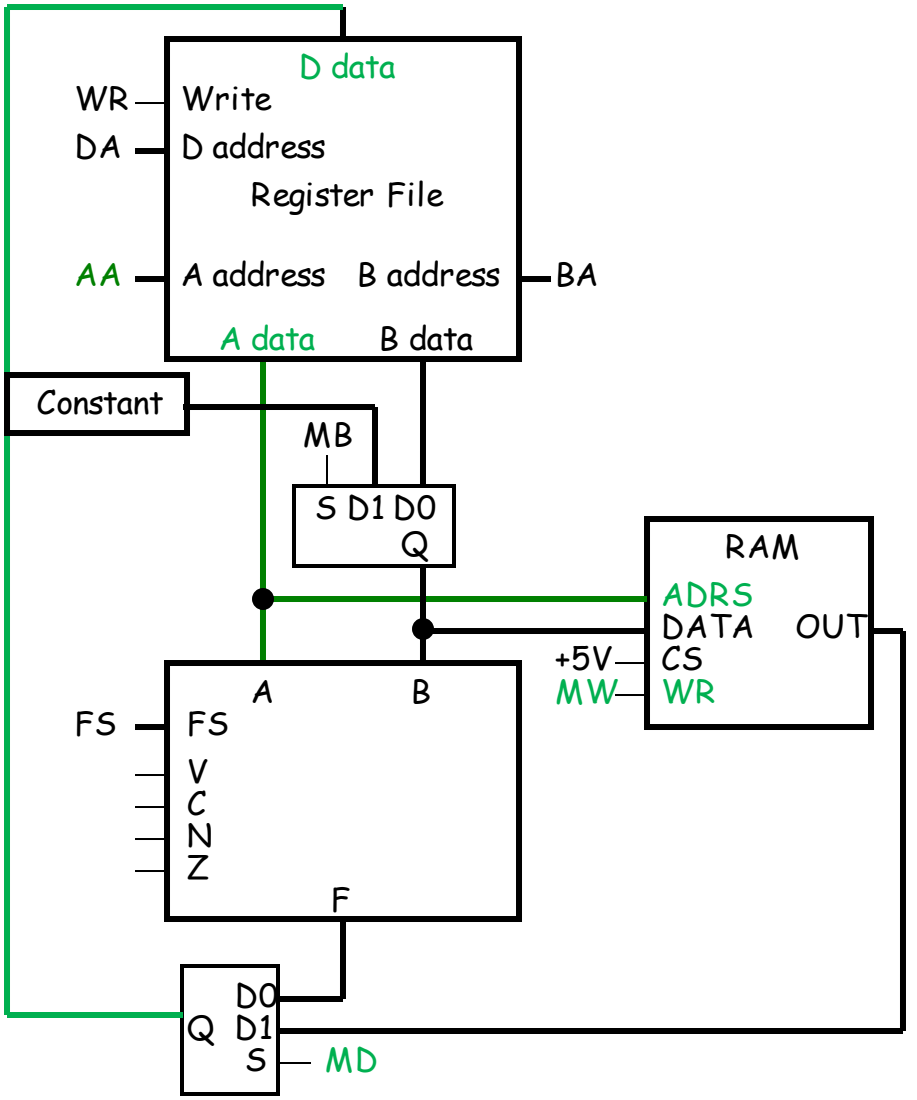
Accessing RAM

- Here's a way to connect RAM into our existing datapath.
- ***Write* to RAM:** we must give an address and a data value.
- These will come from the registers/Constant. We connect A data to the memory's ADRS input, and B data to the memory's DATA input.
- Set MW = 1 to write to the RAM. (It's called MW to distinguish it from the WR write signal on the register file.)
- WR = 0 to prevent any change to register file
- In the next clock cycle B data will be written to A data location



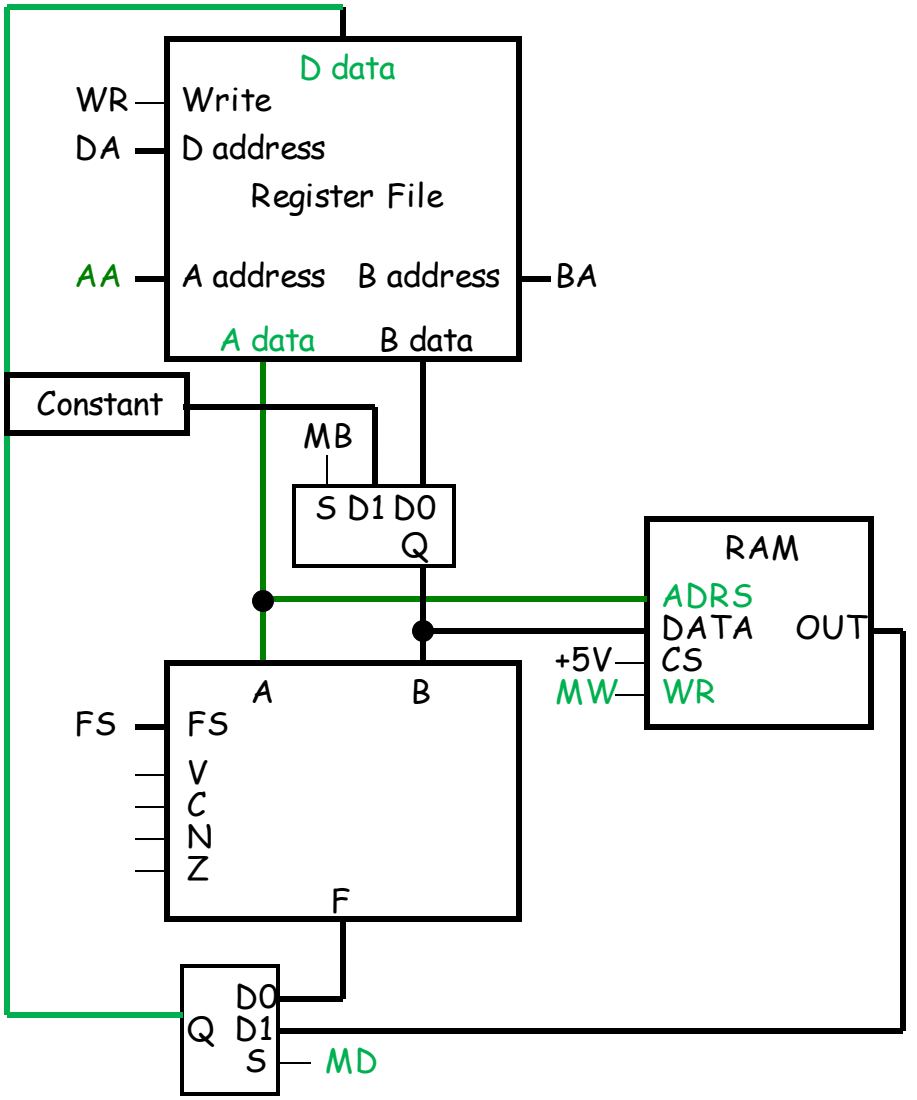
Accessing RAM…

- **Read from RAM**: A data must supply the address.
- Set MW = 0 for reading.
- The incoming data will be sent to the register file for storage.
- This means that the register file's D data input could come from *either* the ALU output or the RAM.
- A mux MD selects the source for the register file.
- When MD = 1, the RAM output is sent to the register file instead.



Notes about this setup …

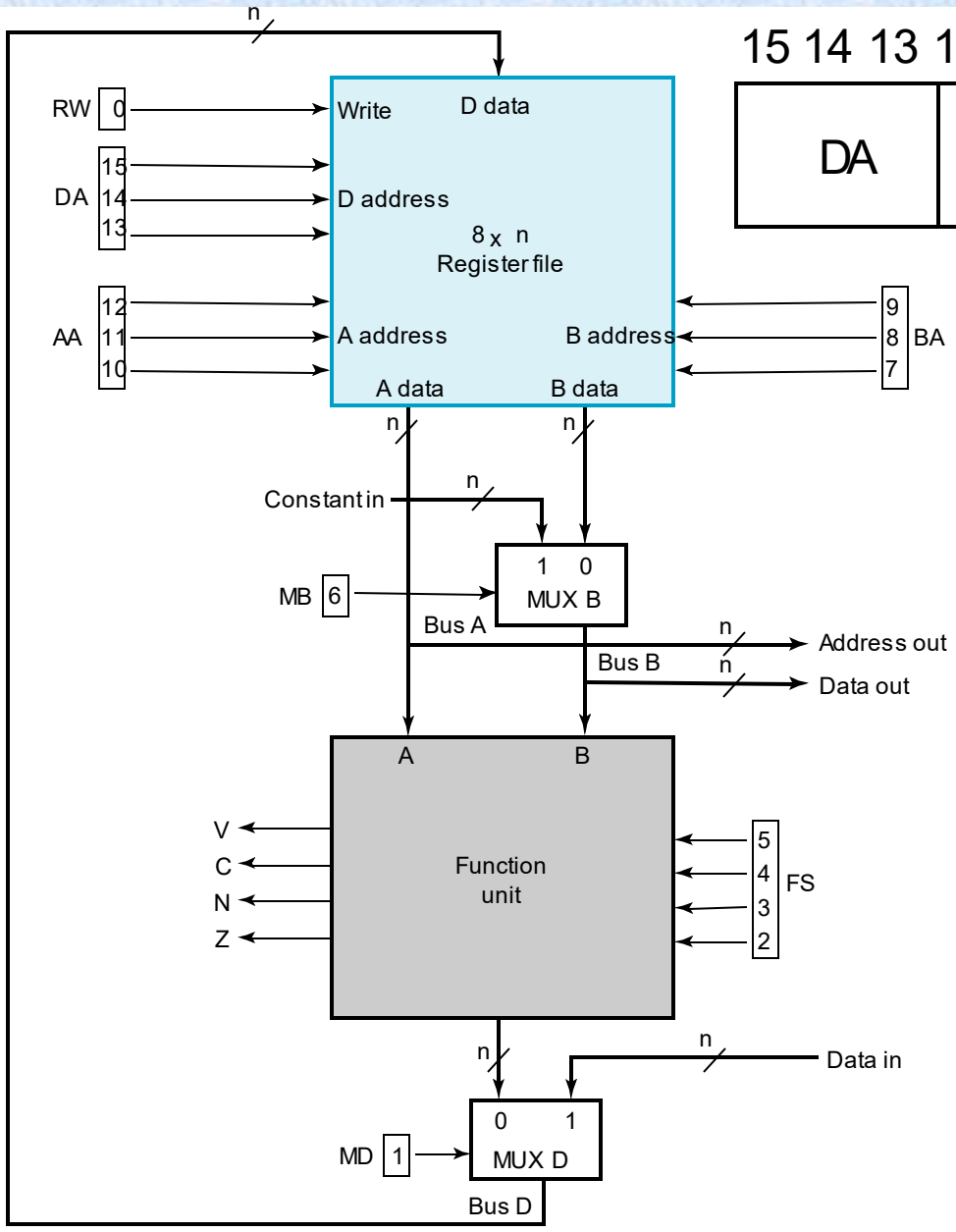
- We now have a way to copy data between our register file and the RAM.
- Notice that there's no way for the Function Unit to directly access the memory—RAM contents *must* go through the register file first.
- Here the size of the memory is limited by the size of the registers; with n-bit registers, we can only use a 2^n x n RAM.

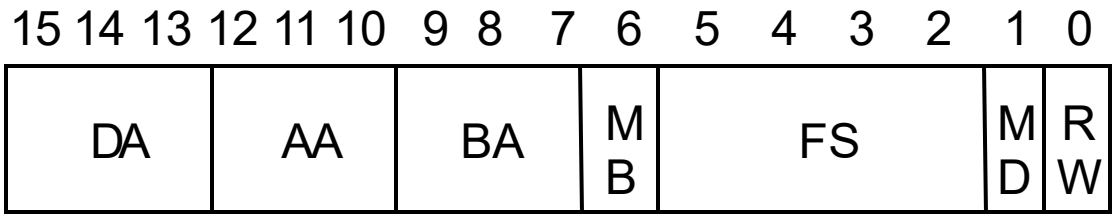


The Control Word

- The datapath has many control inputs
- The signals driving these inputs can be defined and organized into a *control word*
- To execute a microinstruction, we apply control word values for a clock cycle.
- For most microoperations, the positive edge of the clock cycle is needed to perform the register load
- The datapath control word format and the field definitions are shown on the next slide

The Control Word Fields …





Control word

- **The Control Words Fields**
  - DA – D Address (destination)
  - AA – A Address
  - BA – B Address
  - MB – Mux B (constant/source
  - FS – Function Select
  - MD – Mux D
  - RW – Register Write
  - MW – RAM read/write
    (Not shown here)

Encoding The Control Word …

TABLE 8-5
Encoding of Control Word for the Datapath

DA, AA, BA		MB		FS		MD		RW	
Function	Code	Function	Code	Function	Code	Function	Code	Function	Code
$R0$	000	Register	0	$F = A$	0000	Function	0	No Write	0
$R1$	001	Constant	1	$F = A + 1$	0001	Data in	1	Write	1
$R2$	010			$F = A + B$	0010				
$R3$	011			$F = A + B + 1$	0011				
$R4$	100			$F = A + \overline{B}$	0100				
$R5$	101			$F = A + \overline{B} + 1$	0101				
$R6$	110			$F = A - 1$	0110				
$R7$	111			$F = A$	0111				
				$F = A \wedge B$	1000				
				$F = A \vee B$	1001				
				$F = A \oplus B$	1010				
				$F = \overline{A}$	1011				
				$F = B$	1100				
				$F = \text{sr } B$	1101				
				$F = \text{sl } B$	1110				

Microoperations for the Datapath…

- Symbolic Representation

TABLE 8-6
Examples of Microoperations for the Datapath, Using Symbolic Notation

Micro-operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 - R3$	R1	R2	R3	Register	$F = A + \overline{B} + I$	Function	Write
$R4 \leftarrow$ sl R6	R4	_	R6	Register	$F =$ sl B	Function	Write
$R7 \leftarrow R7 + 1$	R7	R7	_	_	$F = A + 1$	Function	Write
$R1 \leftarrow R0 + 2$	R1	R0	_	Constant	$F = A + B$	Function	Write
$M(R1) \leftarrow R3$	_	R1	R3	Register	_	_	No Write
$R4 \leftarrow M(R1)$	R4	R1	_	_	_	RAM	Write
$R5 \leftarrow 0$	R5	R0	R0	Register	$F = A \oplus B$	Function	Write

Microoperations for the Datapath …

- Binary Representation

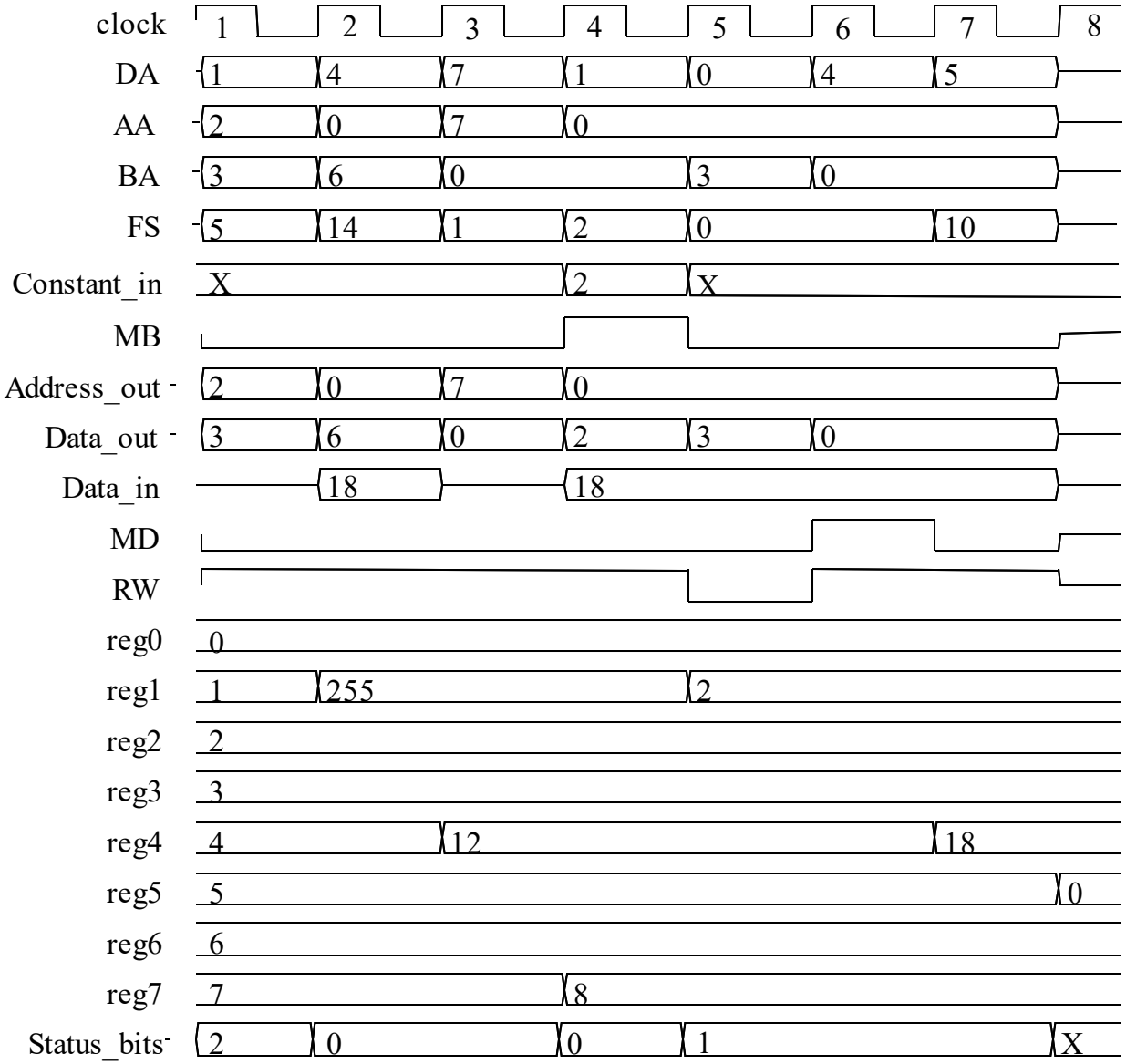
TABLE 8-7
Examples of Microoperations from Table 8-6, Using Binary Control Words

Micro-operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 - R3$	001	010	011	0	0101	0	1
$R4 \leftarrow$ sl R6	100	XXX	110	0	1110	0	1
$R7 \leftarrow R7 + 1$	111	111	XXX	X	0001	0	1
$R1 \leftarrow R0 + 2$	001	000	XXX	1	0010	0	1
$M(R1) \leftarrow R3$	XXX	001	011	0	XXXX	X	0
$R4 \leftarrow M(R1)$	100	001	XXX	X	XXXX	1	1
$R5 \leftarrow 0$	101	000	000	0	1010	0	1

Results of simulation of the above on the next slide

Datapath Simulation …

- **Results of simulation of the above**



Stored Program Computer

- Control words operate the datapath
- Store control words in memory
  - Sequence through them to perform a series of computational steps