Introduction to Digital Circuits and Computer Design

Notes #9

"Computer Design - Control Unit - Part 2"

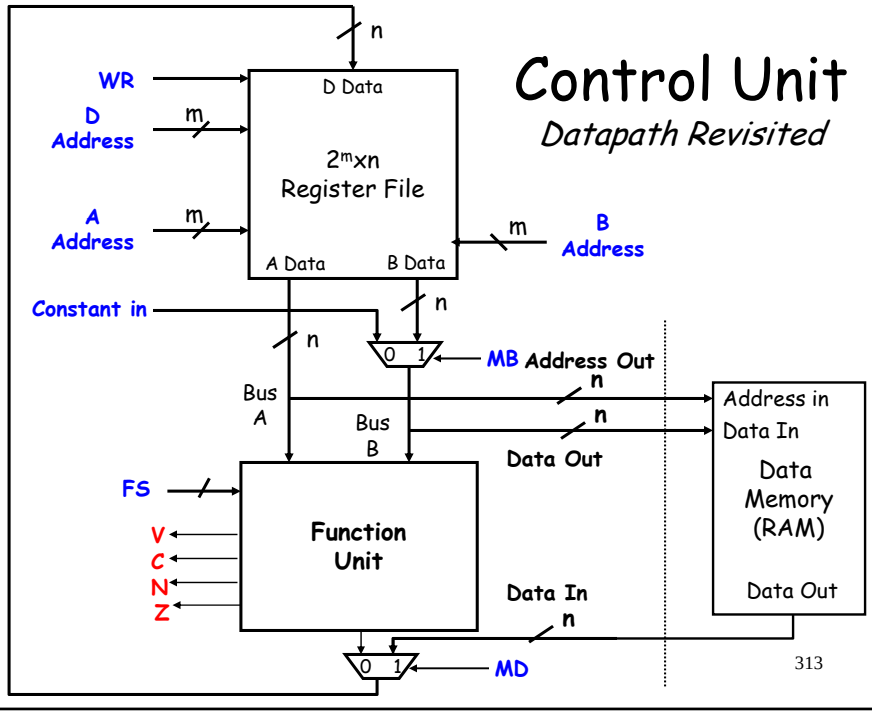
Avinash Kodi

Spring Quarter 2010

Acknowledgement: Dr. Maarten Uijt de Haag

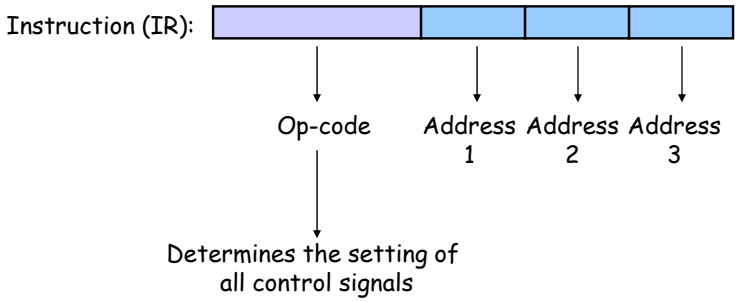
Control Unit

Datapath Revisited



Instruction Interpretation

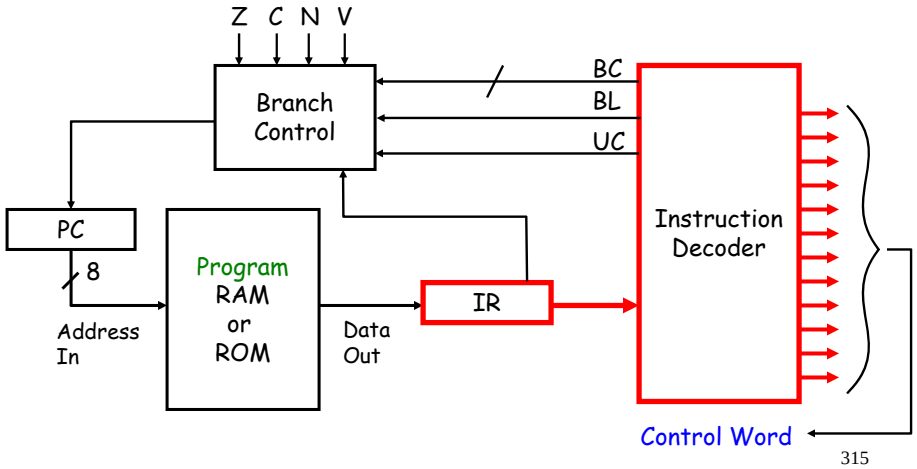
Instruction Decoder - 1



Note: given example is a 3-address instruction

Instruction Interpretation

Instruction Decoder - 2



315

Instruction Sequencing
Decision Making & Branching - 1

Branching / Jumping / Transfer of Control:
refers to a change in the execution path within a program
(jump to a different instruction of the program)

Unconditional branching

Transfer of control or jumping within the program takes place no matter what.

Conditional branching

Transfer of control or jumping takes place if and only if a specific condition is satisfied.

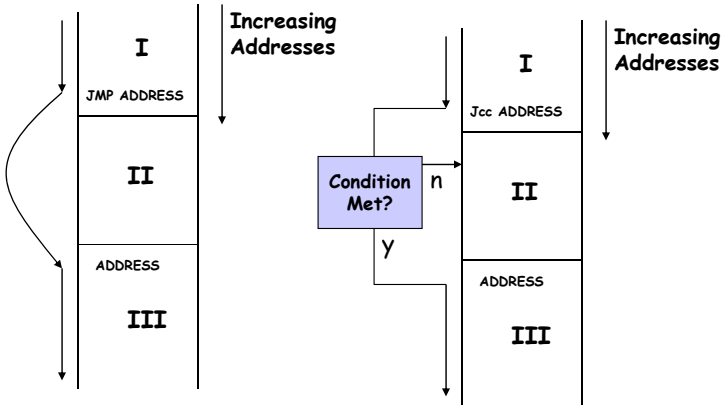
These conditions are often coupled to one or more flags in the status register.

C, Z, N, V

The user program must set up one or more of these flags to be tested by the branch instruction!

316

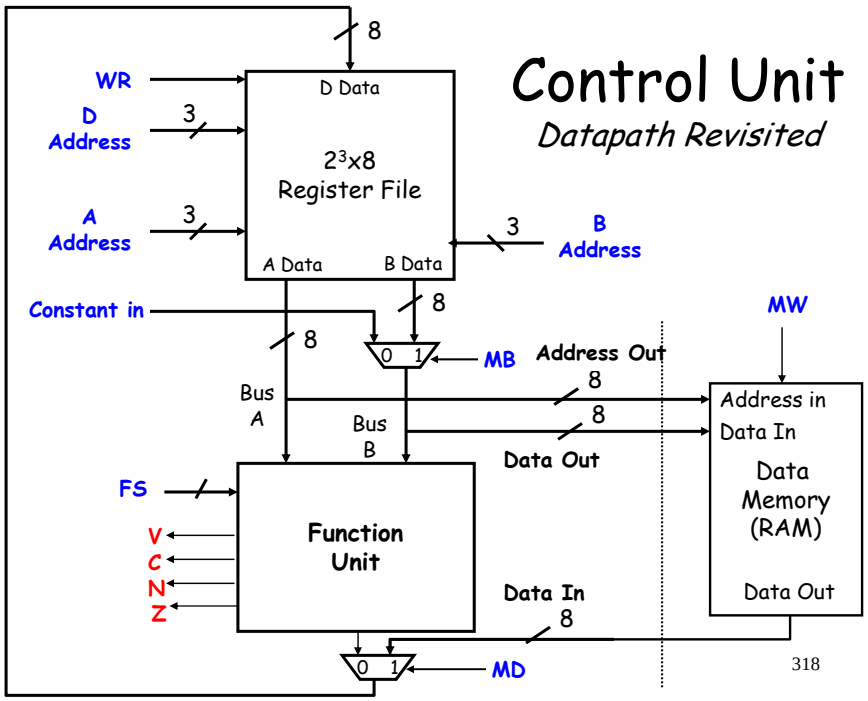
Instruction Sequencing
Decision Making & Branching - 2



317

Control Unit

Datapath Revisited



Datapath Control

What is our Real-Estate?

Registers (DA, AA, BA):		MD :		MB :		WR :	
R1	000	Data from		Data from		No write	0
R2	001	Function Unit	0	Register	0	Write enable	1
R3	010	Data from		Constant from			
R4	011	memory	1	instruction	1		
R5	100						
R6	101						
R7	110						
R8	111						

Requires 3 instruction decoder outputs, or 3 control word bits

Requires 1 instruction decoder output, or 1 control word bit

Requires 1 instruction decoder output, or 1 control word bit

Requires 1 instruction decoder output, or 1 control word bit

Datapath Control
Function Unit

FS:

$F = A$	00000
$F = A + 1$	00001
$F = A + B$	00010
$F = A + \overline{B}$	00011
$F = A + \overline{B} + 1$	00100
$F = A - 1$	00101
$F = A$	00110
$F = A \wedge B$	00111
$F = A \vee B$	01000
$F = A \oplus B$	01010
$F = \overline{A}$	01100
$F = B$	01110
$F = sr\ B$	10000
$F = sl\ B$	10100
	11000

Requires 5 instruction decoder outputs, or 5 control word bits

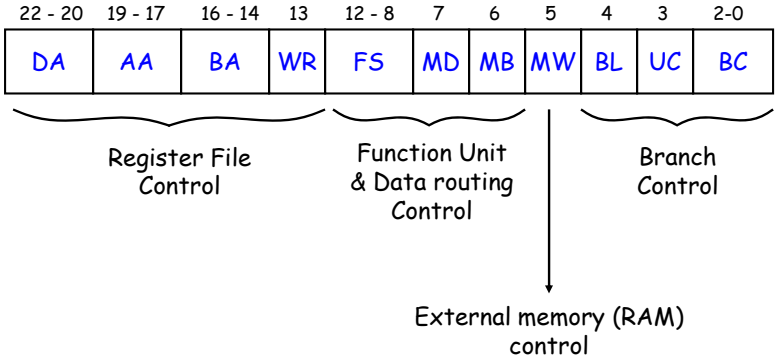
And of course don't forget about the branch control inputs: BC, BL, and UC

BC:

000	C = 1
001	N = 1
010	Z = 1
011	V = 1
100	C = 0
101	N = 0
110	Z = 0
111	V = 0

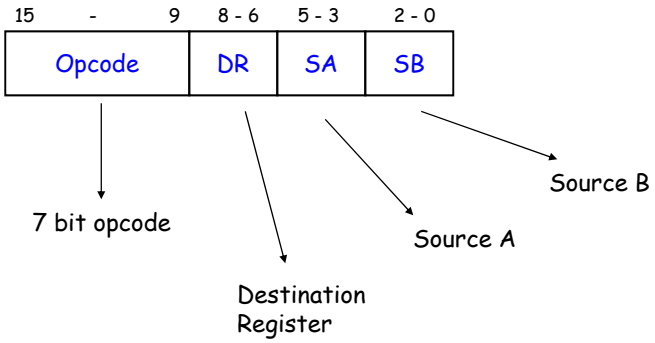
320

Datapath Control
Control Word



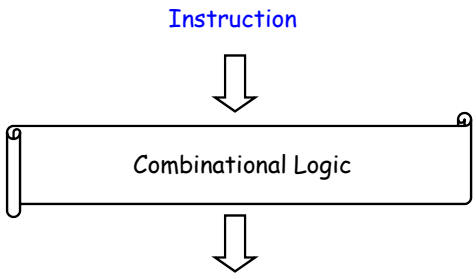
321

Datapath Control
Instruction



322

Control Interpretation
Instruction Decoder -3



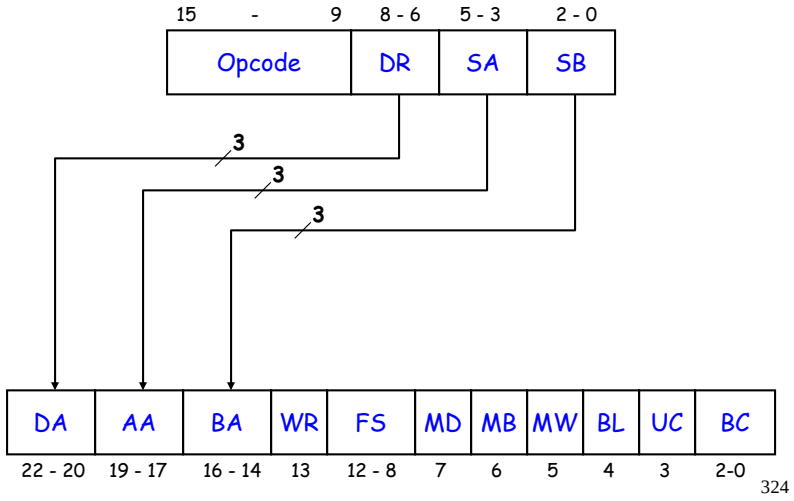
Q: Why not just use the control word as instruction?

Note: **it is not always as simple as finding combinational logic, more often is is necessary to use sequential logic, but more about that later.**

323

Control Unit

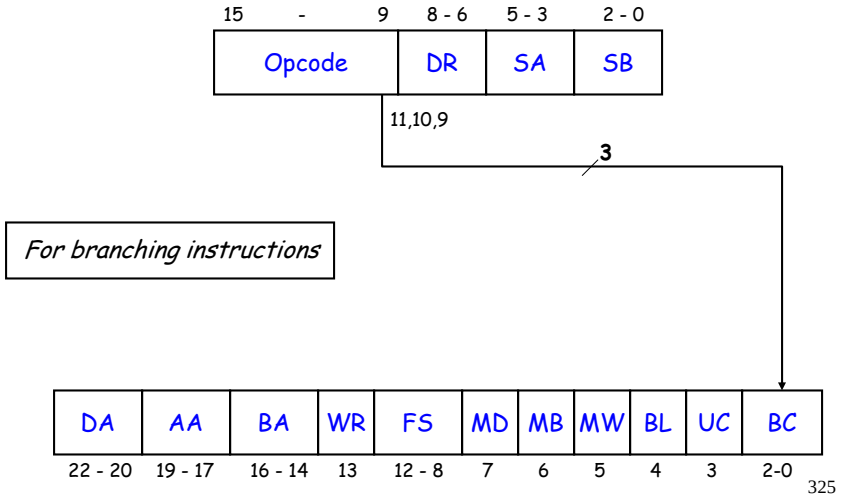
Instruction Interpretation - 1



324

Control Unit

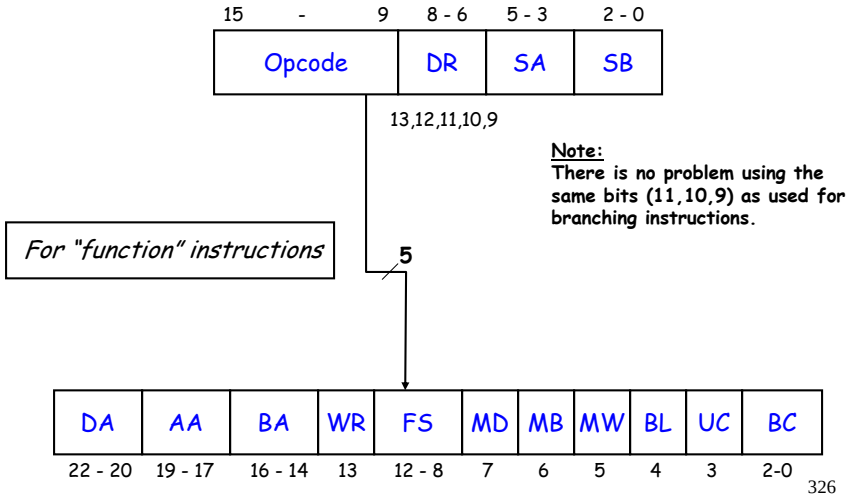
Instruction Interpretation - 2



325

Control Unit

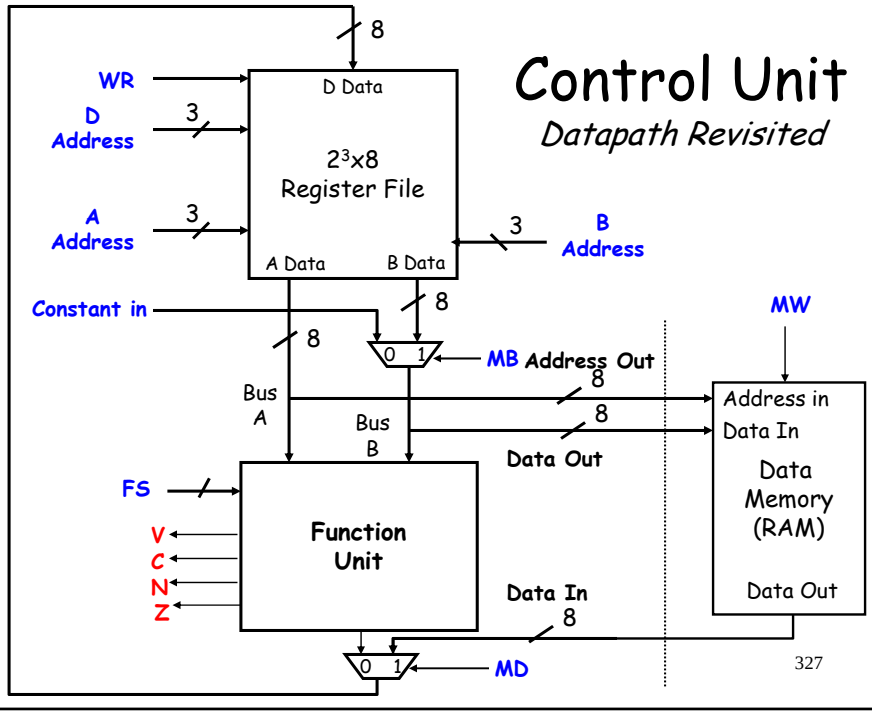
Instruction Interpretation - 3



326

Control Unit

Datapath Revisited



327

Control Unit

Instruction Interpretation - 4

What about the others?

	Instruction Bits			Control Word Bits					
	Bit 15	Bit 14	Bit 13	MB	MD	WR	MW	BL	UC
ALU function with registers	0	0	-	1	0	1	0	0	X
Memory write using register data	0	1	0	1	X	0	1	0	X
Memory read using register data	0	1	1	1	1	1	0	0	X
ALU operation using a constant	1	0	-	0	0	1	0	0	X
Conditional branch	1	1	0	X	X	0	0	1	0
Unconditional branch	1	1	1	X	X	0	0	1	1

MB

15 \ 14,13	00	01	11	10
0	1	1	1	1
1	0	0	X	X

$MB = \overline{Bit15} + Bit14$

MD

15 \ 14,13	00	01	11	10
0	0	0	1	X
1	0	0	X	X

$MD = Bit14$

$WR = \overline{Bit14} + \overline{Bit15} \bullet Bit13$

$MW = \overline{Bit15} \bullet Bit14 \bullet \overline{Bit13}$

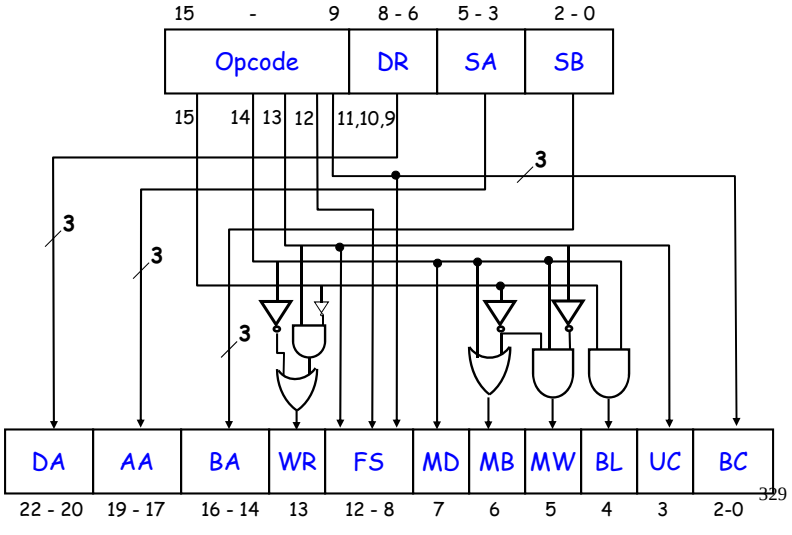
$BL = Bit15 \bullet Bit14$

$UC = Bit13$

328

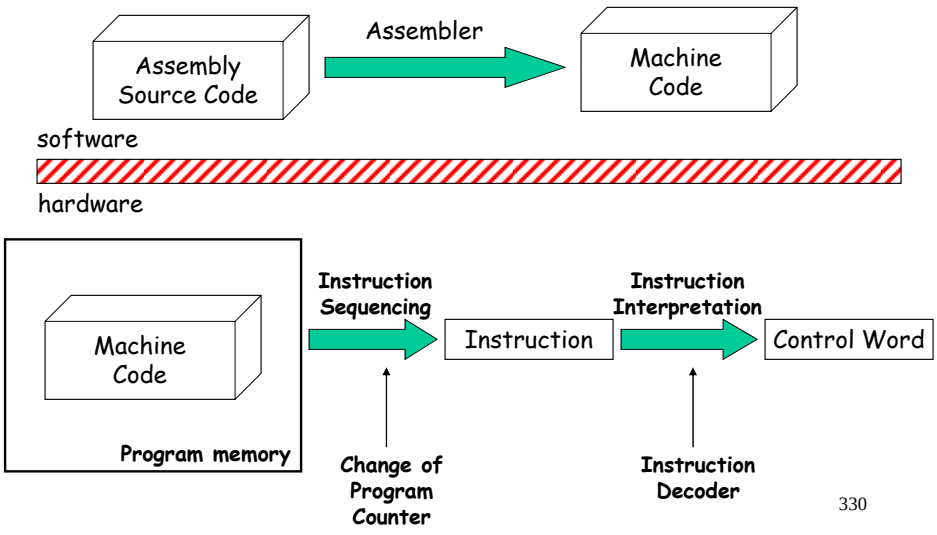
Control Unit

Instruction Interpretation - 5



329

Interpretations



330

Assembly

Mnemonic	Assembly		RTL	Machine Code
				Rx
MOV:	MOV	Rx, [M]	; (Rx) ←(M)	0110000 xxx --- ---
	MOV	[M], Rx	; (M) ←(Rx)	0100000 --- --- xxx
	MOV	Rx, Ry	; (Rx) ←(Ry)	0000000 xxx yyy ---
ADD:	ADD	Rx, Ry	; (Rx) ← (Rx) + (Ry)	0000010 xxx xxx yyy
SUB:	SUB	Rx, Ry	; (Rx) ← (Rx) - (Ry)	0000101 xxx xxx yyy
SHL:	SHL	Rx	; (Rx) ← sl (Rx)	0011000 xxx xxx ---
	etc.		etc.	etc.

undefined or don't care

Note: this is NOT Intel stuff, just an example

331

Timing Considerations
Example 1

Addition:

(Rx) ← (Rx) + (Ry)

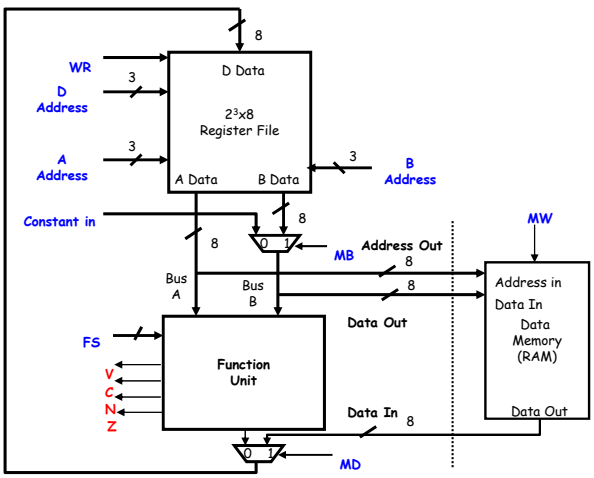
How long does it take to execute this instruction?

Propagation delays:

Read from register file:	3 ns
Write to register file:	3 ns
Read from data memory:	5 ns
Write to data memory:	7 ns
Multiplexer:	1 ns
Function unit:	4 ns

Addition:

Read from register	3ns
Multiplexer	1 ns
Add function	4 ns
Multiplexer	1 ns
Write to register	3 ns
	12 ns



It takes 12 ns to perform this (micro-)operation

Timing Considerations
Example 2

Memory Write:

(M) ← (Rx)

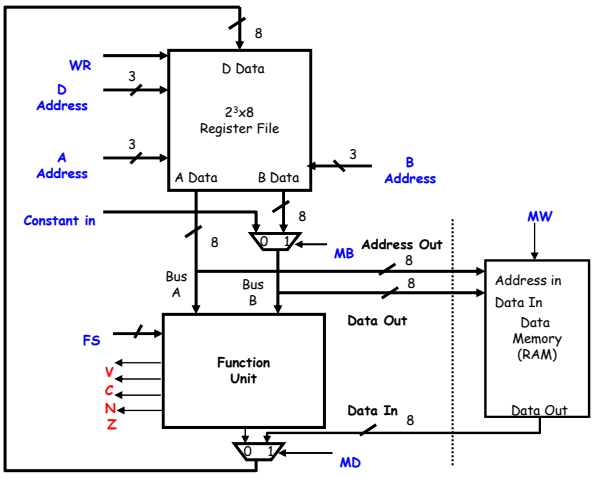
How long does it take to execute this instruction?

Propagation delays:

Read from register file:	3 ns
Write to register file:	3 ns
Read from data memory:	5 ns
Write to data memory:	7 ns
Multiplexer:	1 ns
Function unit:	4 ns

Addition:

Read from register	3ns
Multiplexer	1 ns
Write to memory	7 ns
	11 ns



It takes 11 ns to perform this (micro-)operation

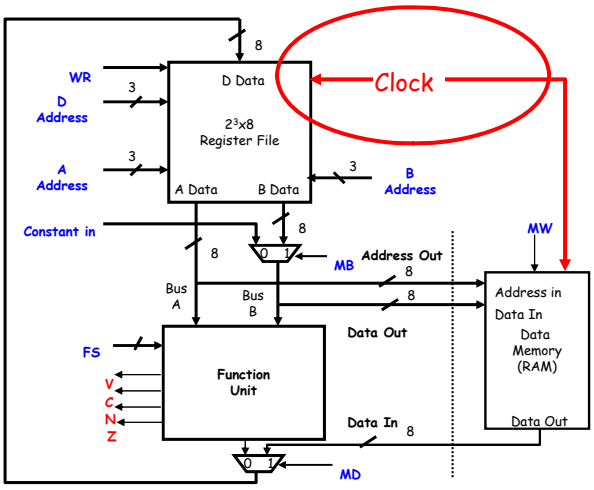
Timing Considerations
Example 3

However, the registers and data memory are clocked !!

What is the maximum clock rate?

Depends on which micro-operation takes the longest time (longest path)!

Here:
Addition takes 12ns, thus absolute maximum clock rate equals:
f_{clk} = 1/12ns ☞ 83MHz



Timing Considerations

In the example every operation took 1 clock cycle.
Because of this we refer to a machine/computer like this as a:

single-cycle implementation

and the control as,

single-cycle control

Disadvantages of single-cycle control:
(or in other words, reason why it is not used in practice)

- inefficient, because every the clock cycle is the same for every instruction; the longest instruction drives the maximum allowable clock speed!

Alternative: Multi-cycle control!!!

Timing & Speed
Some Definitions

CPI	=	Clock-cycles Per Instruction
CPU Clock cycle time	=	how many seconds does one clock cycle last
Instruction count	=	the number of instruction per target program
CPU Execution Time	=	how long does it take to execute a program
	=	(Instruction count) x CPI x (CPU Clock cycle time)

Example: Let's use the previously discussed single-cycle computer
How long does it take to execute a program consisting of 1,200,000 instructions?

Answer: CPI = 1 (one clock cycle per instruction)
Instruction Count = 1,200,000, Clock cycle time = 12ns
CPU execution time = 1,200,000 x 1 x 12ns = 0.0144 seconds

Single-Cycle Machine
Example – Problem Statement

Problem: **Assume the propagation delays of the major components of the processor are:**

- Read from Register file – 4ns
- Write to Register file – 4ns
- Read from data memory – 10ns
- Write to data memory – 12ns
- Read from instruction memory – 3ns
- Multiplexer – 1ns
- Function unit – 3ns

Assume furthermore that the control unit and all other components have no delay!

Assume the following instruction mix for your program:

- 24% load from memory into a register instructions,
- 12% store register contents into memory instruction,
- 44% ALU instructions,
- 18% branch instructions,
- 2% jumps

Question: What is the CPU Clock Cycle Time

Single-Cycle Machine
Example – Solution Part I

							Total
Load from memory into a register	Fetch instruction from instruction memory (3 ns)	Read from data memory (`10ns)	MUX (1ns)	Write to register file (4 ns)			3+10+1+4= 18 ns
Store register to memory	Fetch instruction from instruction memory (3 ns)	Read from register file (4 ns)	MUX (1ns)	Write to Data memory (12ns)			3+4+1+12= 20ns
ALU operation	Fetch instruction from instruction memory (3 ns)	Read from register file (4 ns)	MUX (1ns)	Function Unit (3ns)	MUX (1ns)	Write to register file (4ns)	3+4+1+3+1+4 = 16 ns
Conditional Branch instruction	Fetch instruction from instruction memory (3 ns)						3ns
Unconditional Branch instruction	Fetch instruction from instruction memory (3 ns)						3ns

338

Single-Cycle Machine
Example – Solution Part II

In case of a Single-Cycle machine the CPU Clock Cycle Time is determined by the longest instruction:

CPU Clock Cycle Time = 20ns (Store register to data memory)

Now suppose we do NOT have a single cycle computer but a computer in which the CPU Clock Cycle can vary for each instruction (in other words: each instruction can take whatever time it needs).

Question: What is the **average** CPU Clock Cycle Time of this new computer?

339

Answer

The average CPU Clock Cycle Time can be determined by using the Instruction mix of the program:

Average CPU Clock Cycle Time =

0.24 * 18 ns + 0.12 * 20ns + 0.44 * 16ns + 0.18 * 3ns + 0.02*3ns =

14.36 ns

Faster than the absolute CPU Clock Cycle Time
Of the Single-Cycle computer!!

340