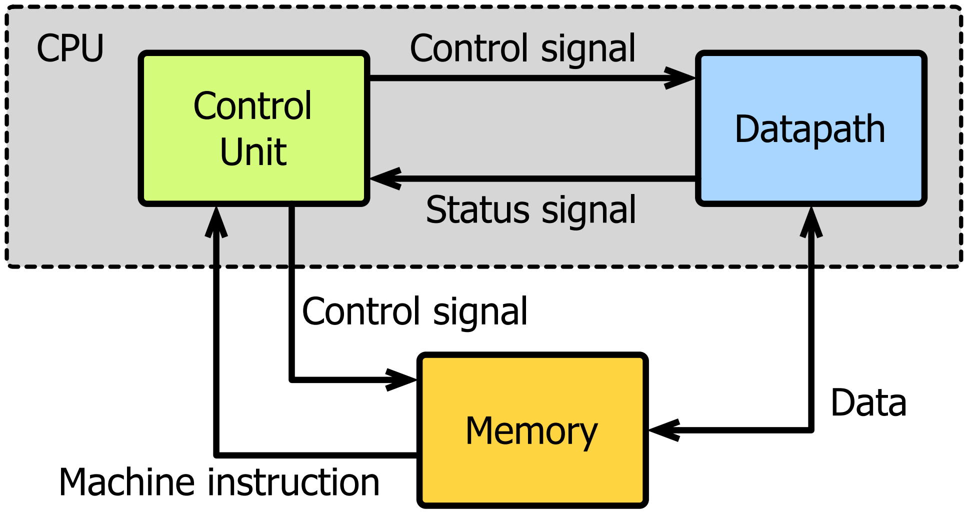
Datapath

010.133
Digital Computer Concept and Practice
Spring 2013

Lecture 05

Digital Systems

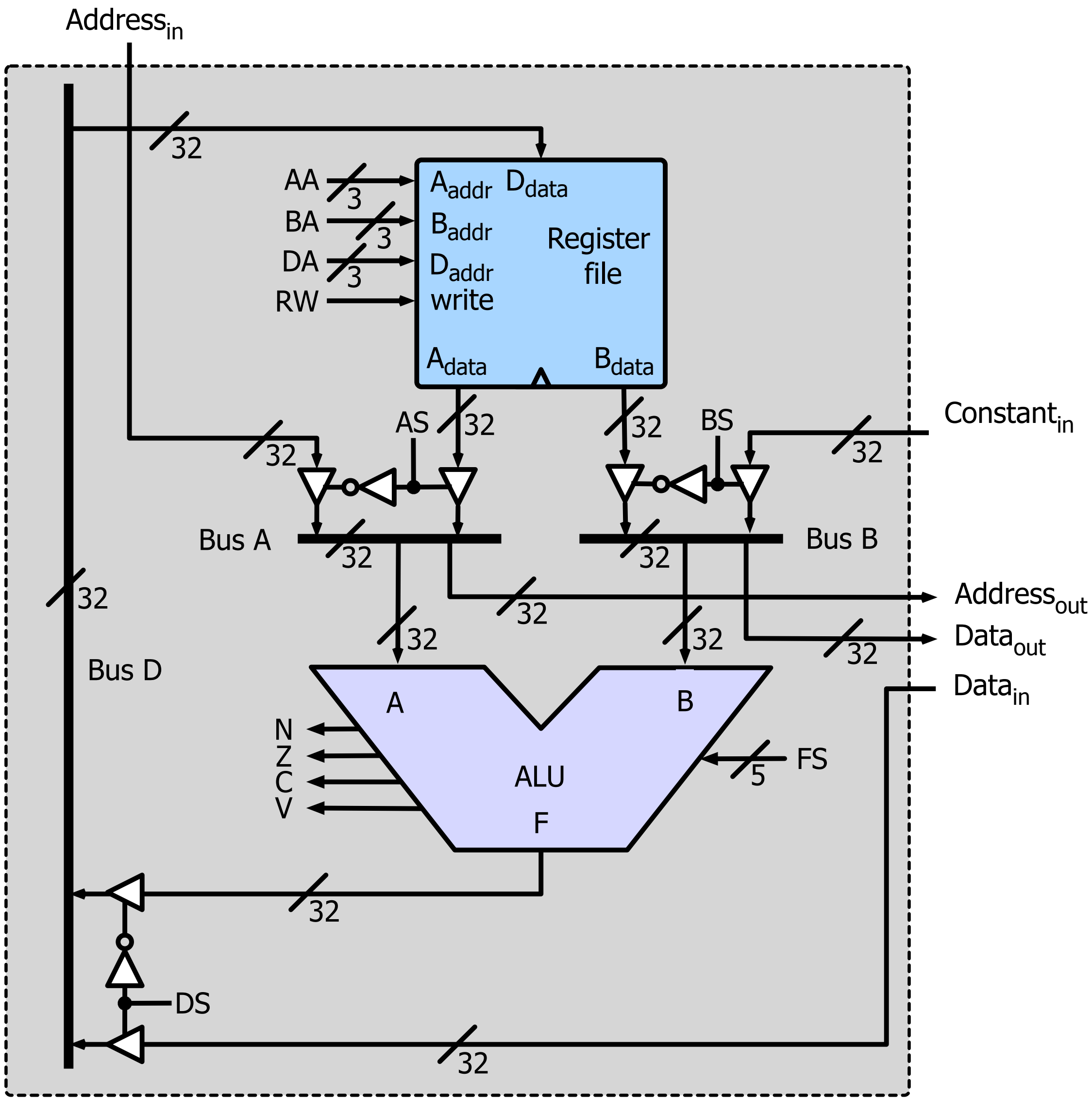
- Digital systems process digital information
  - 0 and 1
  - Datapath + control unit
  - CPUs too
- Datapath
  - A collection of functional units, registers, and interconnections between them that together perform data-processing operations
- Control unit (CU)
  - Controls operations of the datapath and determines the sequence of the operations
  - Coordinates interactions between the datapath and main memory



A 32-bit Datapath

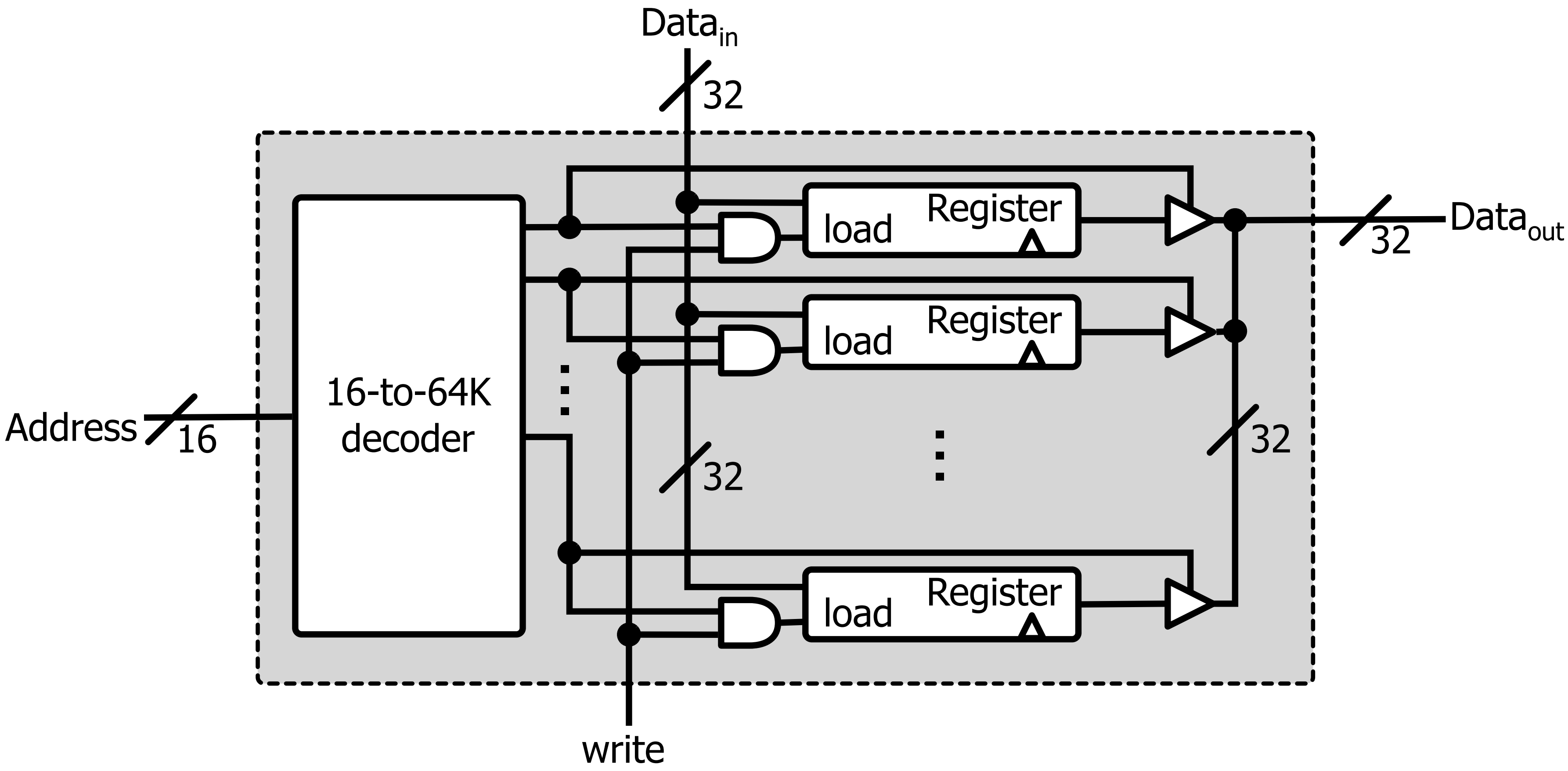
- Register file + ALU

Control bits	Description
AA	specifies a register for the value on the bus A
BA	specifies a register for the value on the bus B
DA	specifies a register to which the value on the bus D is loaded
AS	selects one between the value of a register and $Address_{in}$ for the value on the bus A
BS	selects one between the value of a register and $Constant_{in}$ for the value on the bus B
FS	selects an ALU operation
DS	selects one between the result of the ALU and the data from the memory for the value on the bus D
RW	stores the value on the bus D to the register specified by DA



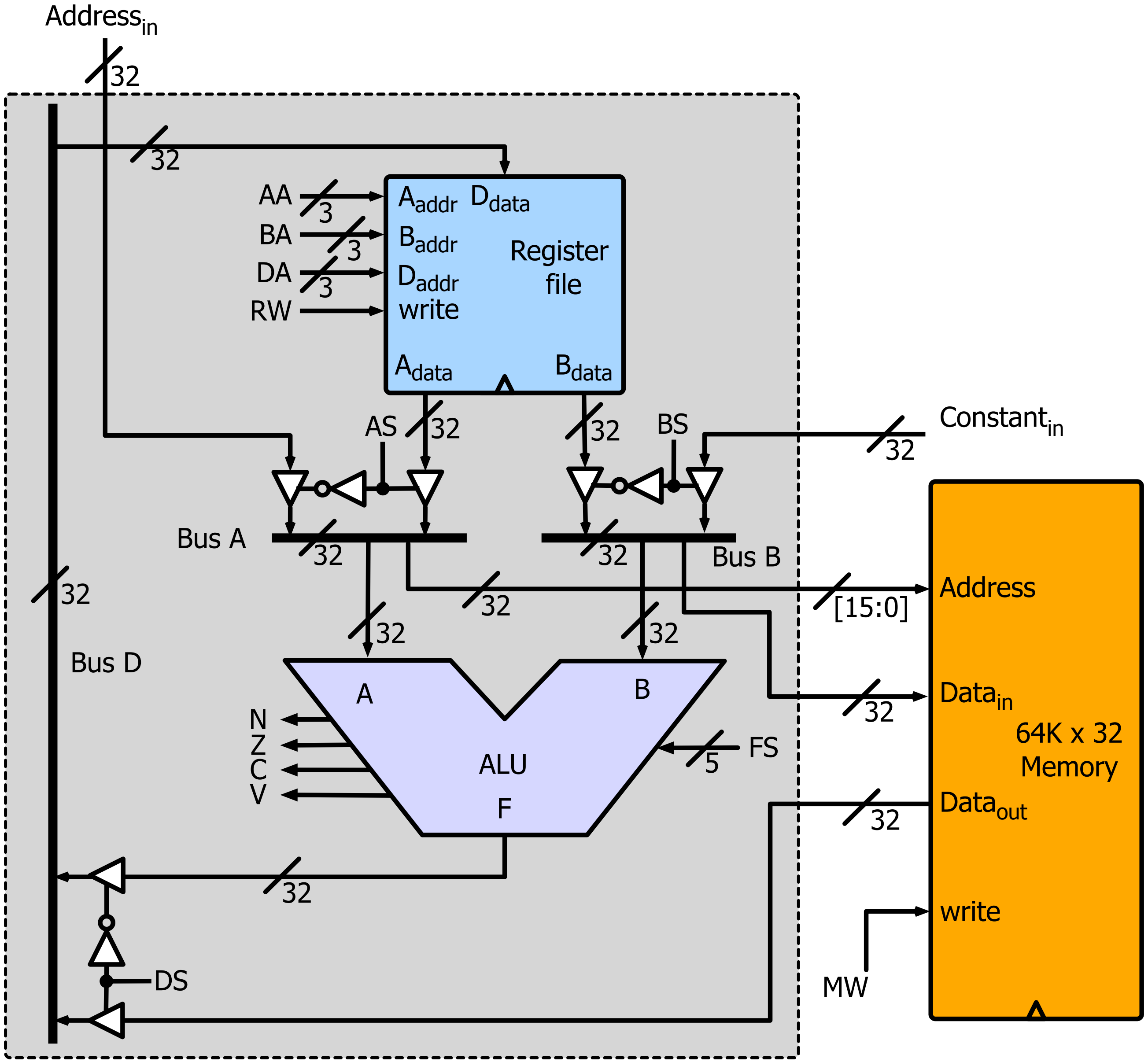
Treating a Memory

- As an array of 2^m n-bit registers
- Write occurs on the positive edge of the next clock cycle
- 64K × 32 memory



Attaching a Memory to the Datapath

- MW
  - stores the value that appears on the bus B into the memory location specified by the address on the bus A



Control Words

- A collection of control bits are called a control word
  - Provided by the control unit in a prescribed manner
  - Its content determines the operation performed in the datapath and memory

18–16	15–13	12–10	9	8	7–3	2	1	0
AA	BA	DA	AS	BS	FS	DS	RW	MW

Microoperations

- The elementary operations on the data stored in the registers
  - Loading a constant value to a register
  - Loading the content of one register to another
  - Adding the contents of two registers and storing the result to another
  - Storing the content of one register in the memory
  - ...
- In our case, each microopration completes in a single clock cycle

Register Transfer Language

- A convenient notation for representing microoperations
- Total 8 registers in the datapath
  - Rx
    - x is the address of the register in the register file
  - R0, R1, …, R7

The Function Table of the ALU

FS_4	FS_3	FS_2	FS_1	FS_0	Operation	
0	0	0	0	0	F = A	Arithmetic unit
0	0	0	0	1	F = A+1	
0	0	0	1	0	F = A+B	
0	0	0	1	1	F = A+B+1	
0	0	1	0	0	F = A+B′	
0	0	1	0	1	F = A+B′+1	
0	0	1	1	0	F = A-1	
0	0	1	1	1	F = A	
0	1	0	0	X	F = $A \vee B$	Logic unit
0	1	0	1	X	F = $A \wedge B$	
0	1	1	0	X	F = $A \oplus B$	
0	1	1	1	X	F = A′	
1	0	0	X	X	F = B	Shifter
1	0	1	0	X	F = logical shift right B	
1	0	1	1	X	F = arithmetic shift right B	
1	1	0	X	X	F = shift left B	
1	1	1	X	X	F = B	

Register Transfer Microoperations

- U ← V
  - U and V are registers (possibly the same)
  - V may be a constant
  - The content of register or a constant V is transferred to the register U
    - The content of U is overwritten on the positive edge of the next clock cycle
    - The content of V remains unchanged

$$R2 \leftarrow R1$$

AA	BA	DA	AS	BS	FS	DS	RW	MW	Constant$_{in}$	Address$_{in}$
0 0 1	X X X	0 1 0	0	X	0 0 0 0 0	0	1	0	0xXXXXXXXX	0xXXXXXXXX

$$R3 \leftarrow 5$$

AA	BA	DA	AS	BS	FS	DS	RW	MW	Constant$_{in}$	Address$_{in}$
X X X	X X X	0 1 1	X	1	1 0 0 X X	0	1	0	0x00000005	0xXXXXXXXX

$$R3 \leftarrow 0$$

AA	BA	DA	AS	BS	FS	DS	RW	MW	Constant$_{in}$	Address$_{in}$
0 0 0	0 0 0	1 0 1	0	0	0 1 1 0 X	0	1	0	0xXXXXXXXX	0xXXXXXXXX

$$R0 \oplus R0 = 0$$

Arithmetic Microoperations

- $U \leftarrow V$ **op$_a$** W
  - U and V are registers, and W may be either of a register and a constant
  - Two or all three of U, V, and W are possibly the same
  - **op$_a$**
    - One of + and -
  - A binary arithmetic operation **op$_a$** is performed on the contents of V and W, and the result is transferred to a register U
    - The content of U is overwritten
    - The contents of V and W remain unchanged

Arithmetic Microoperations (contd.)

$$R0 \leftarrow R1 + R2$$

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant$_{in}$	Address$_{in}$
0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0xXXXXXXXX	0xXXXXXXXX

$$R3 \leftarrow R7 - 4$$

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant$_{in}$	Address$_{in}$
1	1	1	X	X	X	0	1	1	0	1	0	0	1	0	1	0	1	0	0x00000004	0xXXXXXXXX

$$R7 - 4 = R7 + 4' + 1$$

$$R3 \leftarrow R1 + 1$$

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant$_{in}$	Address$_{in}$
0	0	1	X	X	X	0	1	1	0	X	0	0	0	0	1	0	1	0	0xXXXXXXXX	0xXXXXXXXX

$$R3 \leftarrow R1 - 1$$

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant$_{in}$	Address$_{in}$
0	0	1	X	X	X	1	0	1	0	X	0	0	1	1	0	0	1	0	0xXXXXXXXX	0xXXXXXXXX

Logic Microoperations

- $U \leftarrow V$ **op$_l$** W
  - U and V are registers, and W may be either of a register and a constant
  - Two or all three of U, V, and W are possibly the same
  - **op$_l$**
    - $\wedge$, $\vee$, and $\oplus$
  - A binary logic operation **op$_a$** is performed on the contents of V and W, and the result is transferred to a register U
    - The content of U is overwritten
    - The contents of V and W remain unchanged

Logic Microoperations (contd.)

- $U \leftarrow V'$
  - U is a register, and V may be either of a register and a constant
  - U and V are possibly the same
  - The bitwise negation operation is performed on the content of V or a constant V, and the result is transferred to a register U
    - The content of U is overwritten
    - The contents of V remains unchanged

Logic Microoperations (contd.)

$$R0 \leftarrow R1'$$

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant$_{in}$	Address$_{in}$
0	0	1	X	X	X	0	0	0	0	X	0	1	1	1	X	0	1	0	0xXXXXXXXX	0xXXXXXXXX

$$R3 \leftarrow R1 \wedge R2$$

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant$_{in}$	Address$_{in}$
0	0	1	0	1	0	0	1	1	0	0	0	1	0	1	X	0	1	0	0xXXXXXXXX	0xXXXXXXXX

$$R1 \leftarrow R1 \vee R2$$

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant$_{in}$	Address$_{in}$
0	0	1	0	1	0	0	0	1	0	0	0	1	0	0	X	0	1	0	0xXXXXXXXX	0xXXXXXXXX

$$R3 \leftarrow R1 \oplus R2$$

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant$_{in}$	Address$_{in}$
0	0	1	0	1	0	0	1	1	0	0	0	1	1	0	X	0	1	0	0xXXXXXXXX	0xXXXXXXXX

$$R3 \leftarrow R1 \vee \mathrm{0x0F0F0F0F}$$

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant$_{in}$	Address$_{in}$
0	0	1	X	X	X	0	1	1	0	1	0	1	0	0	X	0	1	0	0x0F0F0F0F	0xXXXXXXXX

Shift Microoperations

- U ← V $\mathbf{op_s}$ W
  - U is a register
  - V may be either of a register and a constant
  - $\mathbf{op_s}$
    - lsl (logical shift left)
    - lsr (logical shift right)
    - asr (arithmetic shift right)
  - A 1-bit shift operation $\mathbf{op_s}$ is performed on the content of a register V or a constant V, and the result is transferred to U
    - The content of U is overwritten
    - The content of V remains unchanged

Shift Microoperations (contd.)

$R0 \leftarrow lsl\ R1$

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant_{in}	Address_{in}
0	0	1	X	X	X	0	0	0	0	X	1	1	0	X	X	0	1	0	0xXXXXXXXX	0xXXXXXXXX

$R2 \leftarrow lsr\ R5$

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant_{in}	Address_{in}
1	0	1	X	X	X	0	1	0	0	X	0	1	1	1	X	0	1	0	0xXXXXXXXX	0xXXXXXXXX

$R3 \leftarrow asr\ R7$

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant_{in}	Address_{in}
1	1	1	X	X	X	0	1	1	0	X	1	0	1	1	X	0	1	0	0xXXXXXXXX	0xXXXXXXXX

$R4 \leftarrow asr$ 0x80FF0001

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant_{in}	Address_{in}
X	X	X	X	X	X	1	0	0	X	1	1	0	1	1	X	0	1	0	0x80FF0001	0xXXXXXXXX

Memory Transfer Microoperations

- Memory reads
  - $U \leftarrow M[V]$
    - U and V are possibly the same registers
    - The address of the desired word is given by the content of V
    - The content of U is overwritten
    - The word in the memory remains unchanged

Memory Transfer Microoperations (contd.)

- Memory writes
  - $M[V] \leftarrow U$
    - U and V are possibly the same registers
    - U may be a constant
    - The address of the desired word is the content of V
    - The content of a register U or a constant U is transferred to the memory word specified by V
    - The content of the specified word in the memory is overwritten
    - U remains unchanged

Memory Transfer Microoperations (contd.)

$$R0 \leftarrow M[R1]$$

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant_{in}	Address_{in}
0	0	1	X	X	X	0	0	0	0	X	X	X	X	X	X	1	1	0	0xXXXXXXXX	0xXXXXXXXX

$$M[R2] \leftarrow R4$$

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant_{in}	Address_{in}
0	1	0	1	0	0	X	X	X	0	0	X	X	X	X	X	X	0	1	0xXXXXXXXX	0xXXXXXXXX

$$M[R6] \leftarrow 24$$

AA			BA			DA			AS	BS	FS					DS	RW	MW	Constant_{in}	Address_{in}
1	1	0	X	X	X	X	X	X	0	1	X	X	X	X	X	X	0	1	0x00000018	0xXXXXXXXX

Programming the Datapath

- Summation of 10 numbers from 1 to 10

$$sum = \sum_{i=1}^{10} i$$

- How does a human being perform the summation?

Summation of the 10 Numbers

- R0 contains the result
- It takes 22 clock cycles

$$
\begin{aligned}
&R0 \leftarrow 0 \\
&R1 \leftarrow 0 \\
&R1 \leftarrow R1 + 1 \\
&R0 \leftarrow R0 + R1 \\
&R1 \leftarrow R1 + 1 \\
&R0 \leftarrow R0 + R1 \\
&R1 \leftarrow R1 + 1 \\
&R0 \leftarrow R0 + R1 \\
&R1 \leftarrow R1 + 1 \\
&R0 \leftarrow R0 + R1 \\
&R1 \leftarrow R1 + 1 \\
&R0 \leftarrow R0 + R1 \\
&R1 \leftarrow R1 + 1 \\
&R0 \leftarrow R0 + R1 \\
&R1 \leftarrow R1 + 1 \\
&R0 \leftarrow R0 + R1 \\
&R1 \leftarrow R1 + 1 \\
&R0 \leftarrow R0 + R1 \\
&R1 \leftarrow R1 + 1 \\
&R0 \leftarrow R0 + R1 \\
&R1 \leftarrow R1 + 1 \\
&R0 \leftarrow R0 + R1
\end{aligned}
$$

Summation of Arbitrary 10 Numbers

- The 10 integers are stored in memory consecutively from address 100
- R0 contains the result
- It takes 32 clock cycles
- What if we perform the summation of 10,000 arbitrary numbers?
- The following pattern is repeated:

$$R2 \leftarrow R2 + 1$$
$$R1 \leftarrow M[R2]$$
$$R0 \leftarrow R0 + R1$$

$R0 \leftarrow 0$
$R2 \leftarrow 99$
$R2 \leftarrow R2 + 1$
$R1 \leftarrow M[R2]$
$R0 \leftarrow R0 + R1$
$R2 \leftarrow R2 + 1$
$R1 \leftarrow M[R2]$
$R0 \leftarrow R0 + R1$
$R2 \leftarrow R2 + 1$
$R1 \leftarrow M[R2]$
$R0 \leftarrow R0 + R1$
$R2 \leftarrow R2 + 1$
$R1 \leftarrow M[R2]$
$R0 \leftarrow R0 + R1$
$R2 \leftarrow R2 + 1$
$R1 \leftarrow M[R2]$
$R0 \leftarrow R0 + R1$
$R2 \leftarrow R2 + 1$
$R1 \leftarrow M[R2]$
$R0 \leftarrow R0 + R1$
$R2 \leftarrow R2 + 1$
$R1 \leftarrow M[R2]$
$R0 \leftarrow R0 + R1$
$R2 \leftarrow R2 + 1$
$R1 \leftarrow M[R2]$
$R0 \leftarrow R0 + R1$
$R2 \leftarrow R2 + 1$
$R1 \leftarrow M[R2]$
$R0 \leftarrow R0 + R1$
$R2 \leftarrow R2 + 1$
$R1 \leftarrow M[R2]$
$R0 \leftarrow R0 + R1$

Branching Mechanisms

- A branching mechanism alters control flow
  - Conditional branching
  - Unconditional branching
- Control flow refers to the order in which the individual statements are executed

C Variables and Assignment Operations

- Assignment operations
  - Assigns the value of the right-hand operand to the storage location named by the left-hand operand
  - For example, x = y + 3
- Variables
  - A named storage location that contains some value
  - The variable name typically references the stored value
    - If it is the left-operand of an assignment operation, it refers to the storage location

C Statements and Arrays

- A C statement is a C expression delimited by a semicolon (;)
- An array is a collection of elements that have the same type under the same name
  - An array of n elements
    - Each array element can be treated as a variable and is referenced by the array name and an index number ranging from 0 to n-1
    - a[i] refers to the value of the ith element in the array a
    - A consecutive group of n words in the memory is allocated to the array

The C Code that Adds 10 Arbitrary Integers

```
sum = 0;
i = -1;
i = i + 1;
sum = sum + a[i];
i = i + 1;
sum = sum + a[i];
i = i + 1;
sum = sum + a[i];
i = i + 1;
sum = sum + a[i];
i = i + 1;
sum = sum + a[i];
i = i + 1;
sum = sum + a[i];
i = i + 1;
sum = sum + a[i];
i = i + 1;
sum = sum + a[i];
i = i + 1;
sum = sum + a[i];
i = i + 1;
sum = sum + a[i];
```

```
    sum = 0;
    i = -1;
L:  i = i + 1;
    sum = sum + a[i];
    if (i < 10) goto L;
```

Correspondence between the C and RTL Statements

- Variables sum and i
  - sum ↔ R0
  - i ↔ R2
- 100 is the address of the first element a[0] of array a in the memory

C statement	RTL statements
`sum = 0;`	$R0 \leftarrow 0$
`i = -1;`	$R2 \leftarrow 99$
`i = i + 1;`	$R2 \leftarrow R2 + 1$
`sum = sum + a[i];`	$R1 \leftarrow M[R2]$ $R0 \leftarrow R0 + R1$