

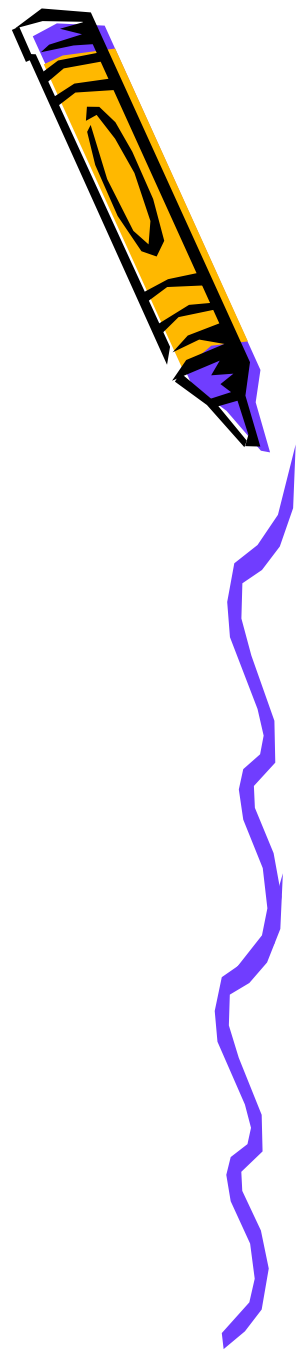


Computer Architecture

Hossein Asadi
Department of Computer Engineering
Sharif University of Technology
asadi@sharif.edu

Today's Topics

- Register Transfer Language (RTL)



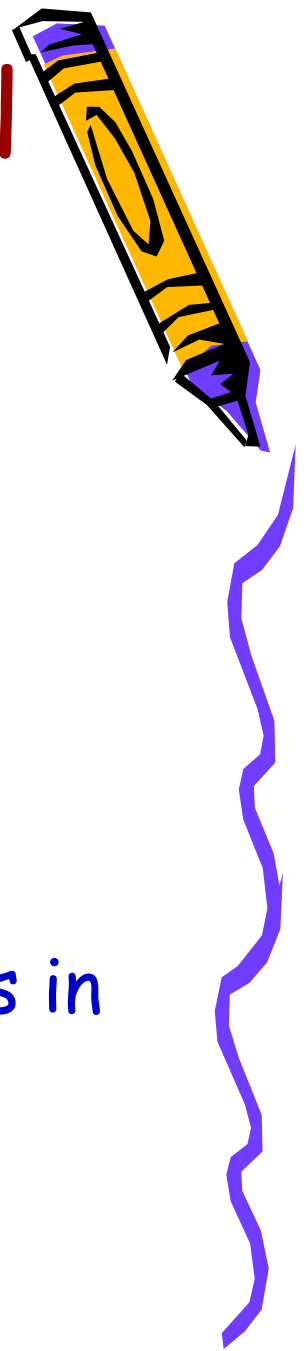
Copyright Notice



- Parts (text & figures) of this lecture adopted from:
 - Computer Organization & Design, The Hardware/Software Interface, 3rd Edition, by D. Patterson and J. Hennessey, MK publishing, 2005.
 - "Intro to Computer Architecture" handouts, by Prof. Hoe, CMU, Spring 2009.
 - "Computer Architecture & Engineering" handouts, by Prof. Kubiawicz, UC Berkeley, Spring 2004.
 - "Intro to Computer Architecture" handouts, by Prof. Hoe, UWisc, Spring 2019.
 - "Computer Arch I" handouts, by Prof. Garzarán, UIUC, Spring 2009.



Data Movements in Digital Systems



- Digital Systems Consists of
 - Combinational logic
 - AND gate, OR gate, NOT gate, etc
 - Sequential elements
 - FFs, latches
- Question:
 - How we can describe data movements in digital systems?
 - Gate-level?
 - Too much details and very complicated



RTL & Micro-Operation



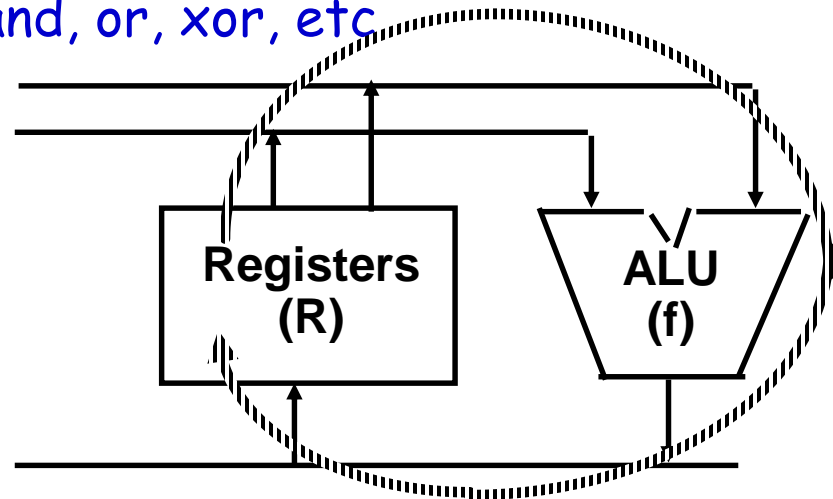
- Data Movement Characterized in terms of:
 - Registers
 - Set of flip-flops/latches
 - Operations done on registers
- Register Transfer Language (RTL)
 - Data movement at register level
 - A language to describe behavior of computers in register flow format
- Micro-operation (Micro-ops or uOps)
 - Functions performed on registers
 - Shift, clear, load, increment, etc.



Micro-Operation



- Definition:
 - An elementary operation performed on information stored in one or more registers
 - During one clock pulse
 - $R \leftarrow F(R, R)$
 - F: shift, load, clear, and, or, xor, etc



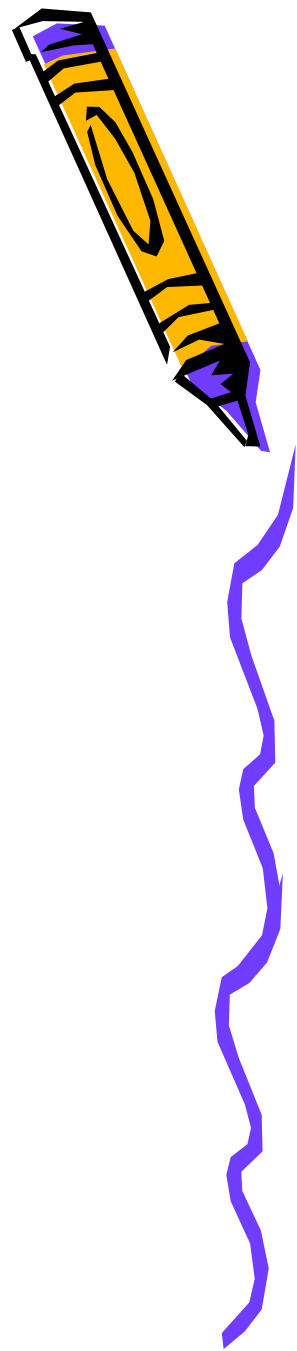
High-Level Description of Computer uArch



- In Terms of RTL & uOps:
 - Set of registers and their functions
 - Microoperations set
 - Set of allowable uops provided by uArch
 - Control signals that initiate sequence of uOps to perform functions



Register Transfer Language



- Our Focus in RTL
 - System's registers
 - Data transfers between them
 - Data transformations in them



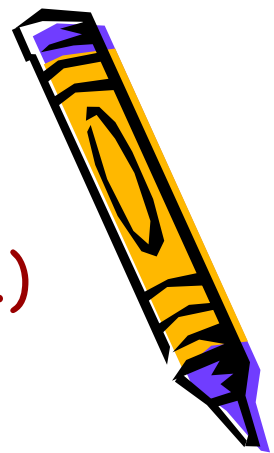
Designation of Registers



- How Registers Designated?
 - Using capital letters
 - Sometimes followed by numbers
 - e.g., A, R13, IR
- Often names indicate function:
 - PC
 - Program Counter
 - IR
 - Instruction Register
 - MAR
 - Memory Address Register



Designation of Registers (cont.)



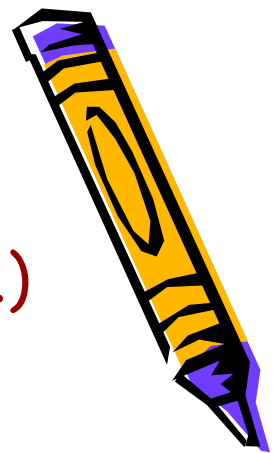
- How Registers Represented?
 - Registers and their contents can be viewed and represented in *various ways*
 - A register can be viewed as a single entity:



- Registers may also be represented showing bits of data they contain



Designation of Registers (cont.)



- How Registers Represented?
 - Common ways of drawing the block diagram of a register

Register



15 0



Numbering of bits

Showing individual bits



15 8 7 0



Subfields



Data Transfer



- Data Transfer:
 - Copying contents of one reg to another reg
- Example:
 - $R2 \leftarrow R1$
 - Contents of R1 copied (loaded) into R2
 - A **simultaneous** transfer from R1 to R2
 - All bits transferred at same time on **one clock cycle**
 - R1: source, R2: destination
 - Data lines from R1 to R2
 - This is a **non-destructive**
 - Contents of R1 not altered by copying them to R2
 - Control lines to perform action



Data Transfer (cont.)



- Controlled (Conditional) Data Transfer
 - Actions will occur if a certain condition is true
 - Similar to an "if" statement in SW programming
 - Represented as:

$P: R2 \leftarrow R1$

"if $P = 1$, then load contents of $R1$ into $R2$ ",
i.e., if $(P = 1)$ then $(R2 \leftarrow R1)$



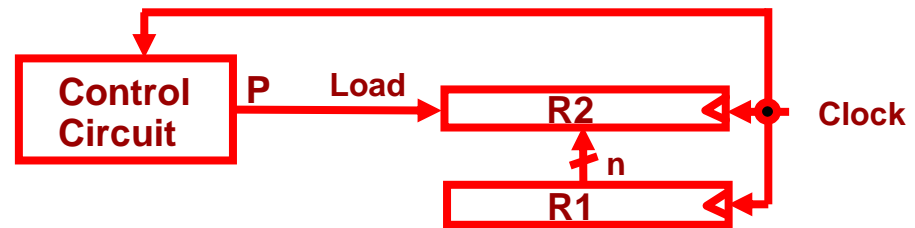
Controlled Data Transfer



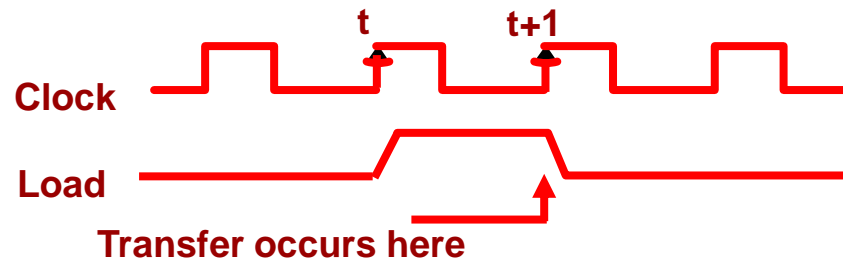
- HW Implementation

P: $R2 \leftarrow R1$

Block diagram



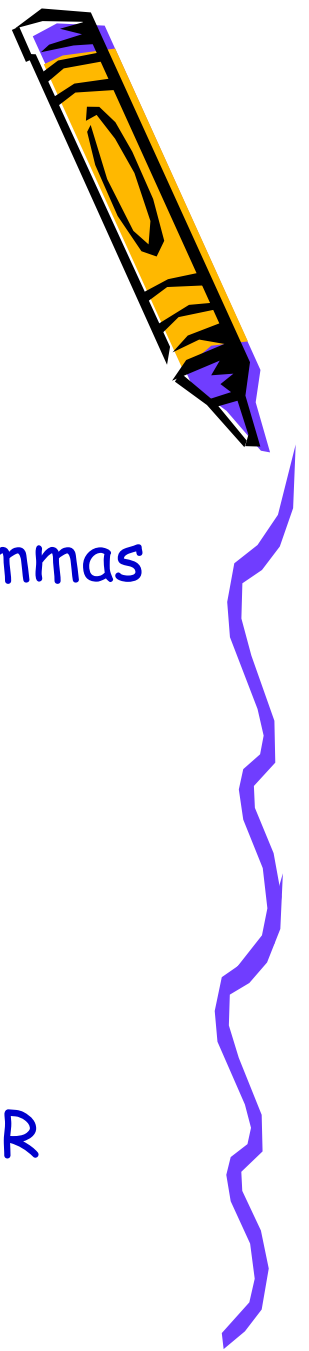
Timing diagram



- Registers assumed to use *positive-edge-triggered* flip-flops



Data Transfer (cont.)



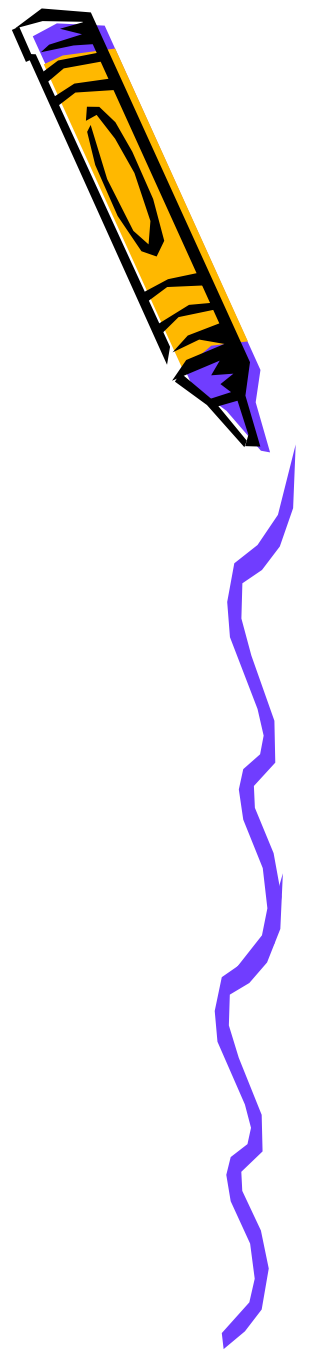
- Simultaneous Micro-operations
 - If two or more operations are to occur simultaneously, they are separated with commas

P: $R3 \leftarrow R5, MAR \leftarrow IR$

- Here, if control function $P = 1 \rightarrow$
 - Load contents of R5 into R3
 - At same clock, load contents of IR into MAR



Data Transfer (cont.)



- Practice
 - Draw hardware implementation of following statement
 - P: $R1 \leftarrow R2, R2 \leftarrow R1$



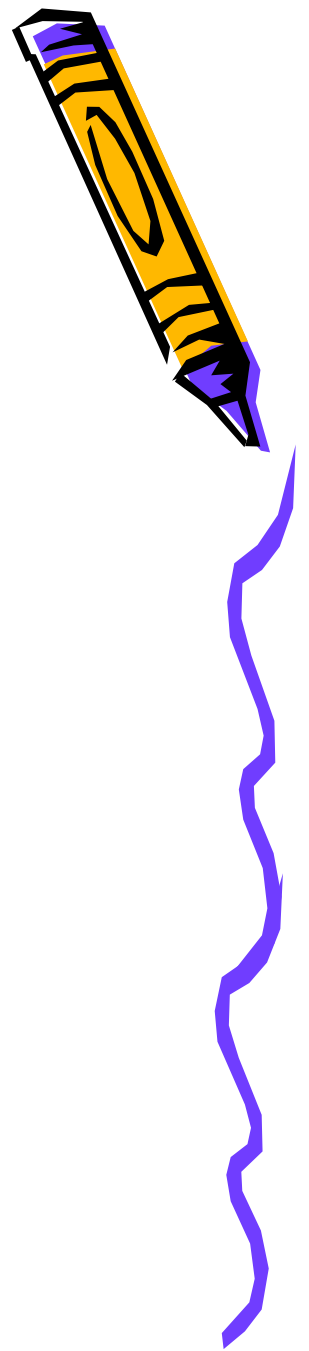
Blocking and Non-Blocking Statements in RTL



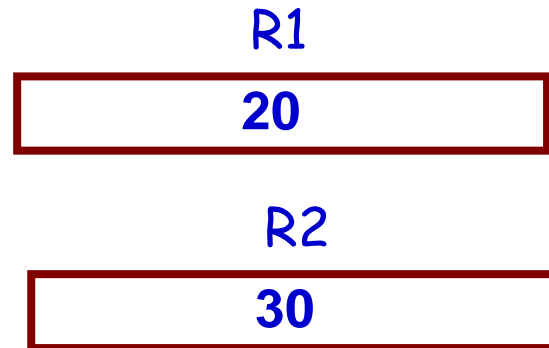
- Non-Blocking Statements
 - Statement scheduled and executed together with other non-blocking assignments
 - Example in Verilog
 - $A \leq B$
 - $B \leq C$
- Blocking Statements
 - Statement executed sequentially
 - Example in Verilog
 - $A = B$
 - $B = C$



Practice



- Code1
 - $R1 \leftarrow R2, R2 \leftarrow R1$
- Code2
 - $R1 \leftarrow R2$
 - $R2 \leftarrow R1$
- Code3
 - $R1 \leftarrow R2$
 - $R2 \leftarrow R1$
- Question:
 - New values of R1 and R2?



Common Micro-Ops



- **Transfer** ($R0 \leftarrow R1$)
 - Transfers data from a reg to another reg
- **Arithmetic** ($R0 \leftarrow R1 + R2$)
 - Performs arithmetic on data in registers
- **Logic/Bit Manipulation**
 - Performs bit (Boolean) operations on data
 $R0 \leftarrow R1 \& R2$; or $R0 \leftarrow R1 | R2$
- **Shift**
 - Shift data in regs by n bits positions
 $R0 \leftarrow R1 \ll 3$; or $R0 \leftarrow R2 \gg 2$



RTL & MIPS Assembly



MIPS Code	RTL Code
add \$t0, \$s2, \$s4	$\$t0 \leftarrow \$s2 + \$s4$
and r1, r2, r3	$r1 \leftarrow r2 \wedge r3$
lw \$s1, 100(\$s2)	$\$s1 \leftarrow \text{Mem}[\$s2 + 100]$
sw \$s1, 100(\$s2)	$\text{Mem}[\$s2 + 100] \leftarrow \$s1$



Summary of Reg Transfer & Micro-Ops



$A \leftarrow B$

$AR \leftarrow DR(AD)$

$A \leftarrow \text{constant}$

$ABUS \leftarrow R1,$

$R2 \leftarrow ABUS$

AR

DR

$M[R]$

M

$DR \leftarrow M$

$M \leftarrow DR$

Transfer content of reg. B into reg. A

Transfer content of AD portion of reg. DR into reg. AR

Transfer a binary constant into reg. A

Transfer content of R1 into bus A and, at the same time,

Transfer content of bus A into R2

Address register

Data register

Memory word specified by reg. R

Equivalent to $M[AR]$

Memory *read* operation: transfers content of memory word specified by AR into DR

Memory *write* operation: transfers content of DR into memory word specified by AR



Summary of Reg Transfer & Micro-Ops (cont.)



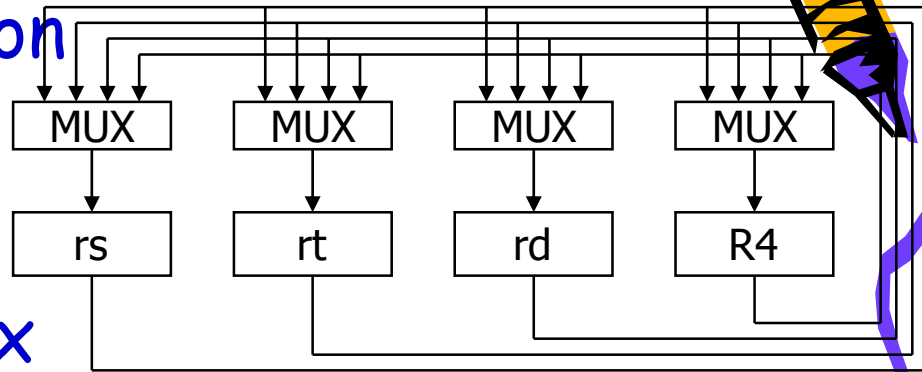
- $GPR[rs]$
 - General Purpose Registers
 - A register from Register File indicated by index rs
- $RF[rs]$
 - Register File



Register Transfers: Interconnect

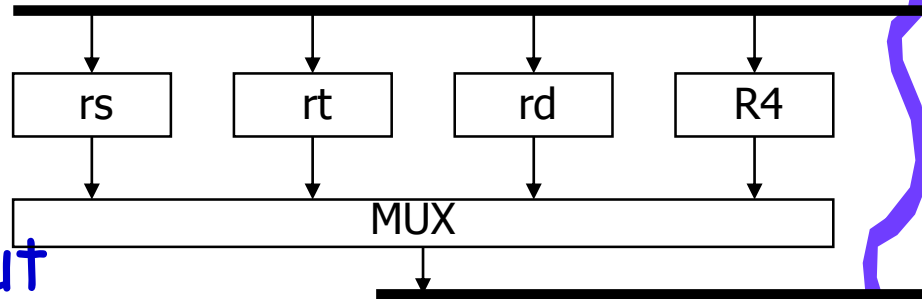
- Point-to-Point Connection

- Dedicated wires
- Muxes on inputs of each register



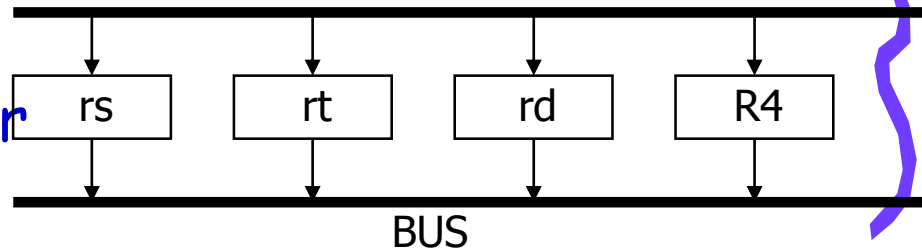
- Common Input from mux

- Load enables for each register
- Control signals for multiplexer



- Common Bus with Output Enables

- Output enables and load enables for each register



Register Transfer: Multiple Busses



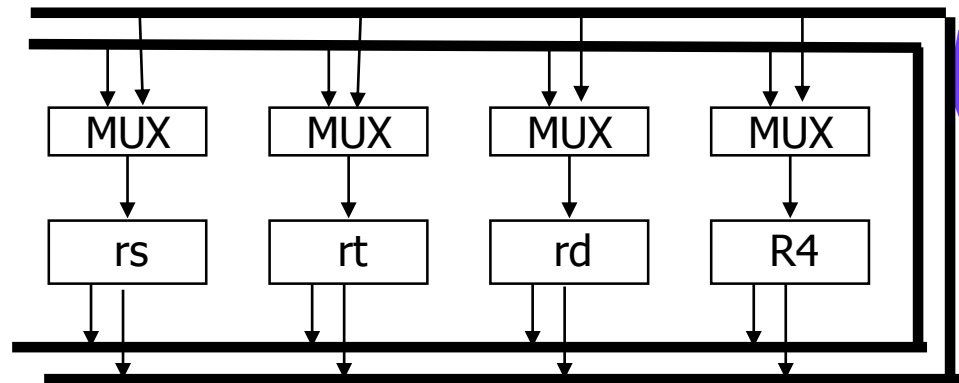
- One transfer per bus
- Each set of wires can carry one value

- State Elements

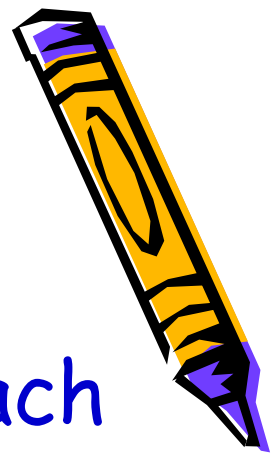
- Registers
- Register files
- Memory

- Combinational Elements

- Busses
- ALUs
- Memory (read)



Connecting Registers

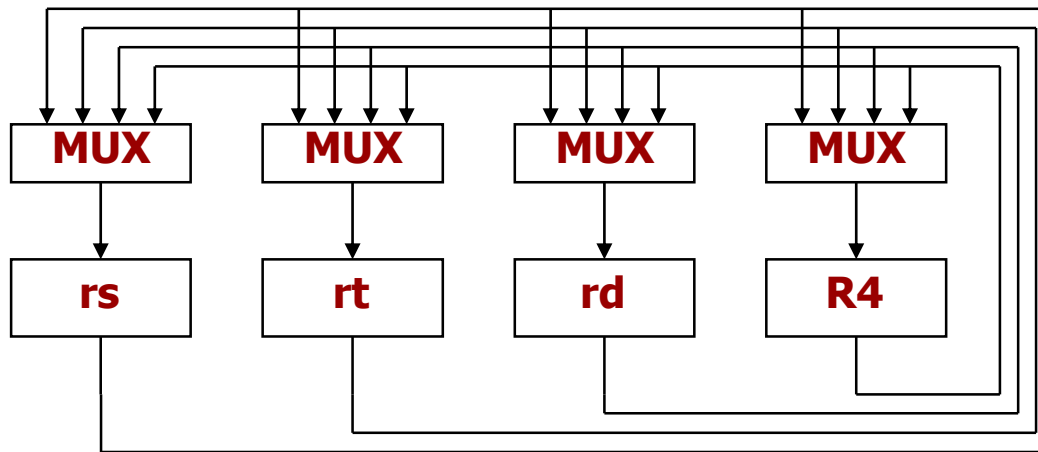


- How to Connect 32 Registers to each other?
 - Impractical to have data and control lines to directly allow each register to be loaded with contents of every possible other registers
- To completely connect n registers
 - $\rightarrow n(n-1)$ lines
 - $O(n^2)$ cost
 - Not a realistic approach to use in a large system

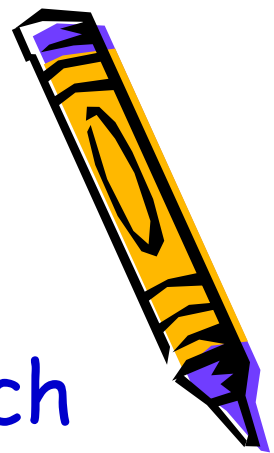


Connecting Registers

- Point-to-Point Connection
 - Dedicated wires
 - Muxes on inputs of each register



Connecting Registers (cont.)

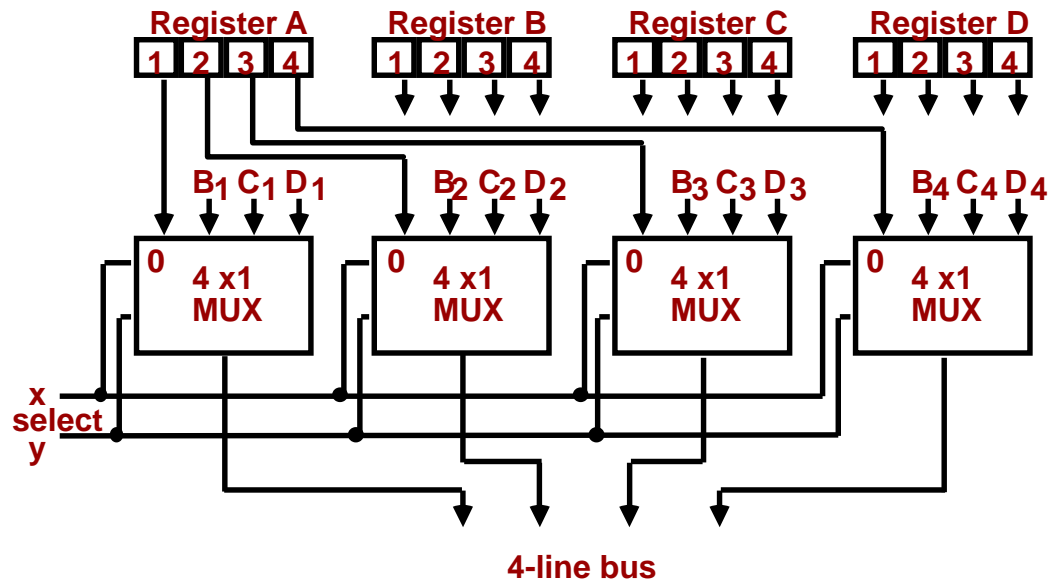
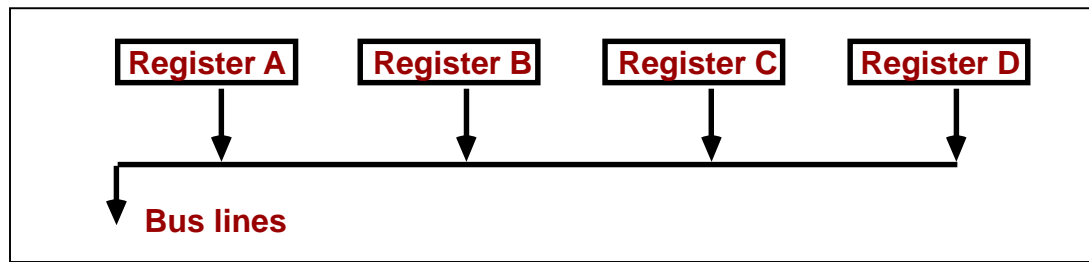


- Instead, Take a Different Approach
 - One centralized set of circuits for data transfer, called **bus**
 - Have control circuits to select:
 - Which reg as source
 - Which reg as destination
- Bus Definition:
 - A path (of a group of wires) over which information is transferred, from any of several sources to any of several destinations
 - A group of wires with multiple drivers

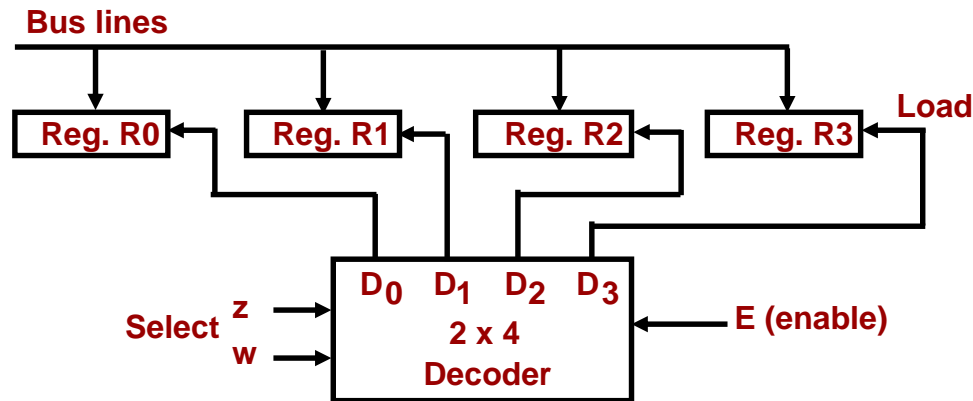


Bus Transfer

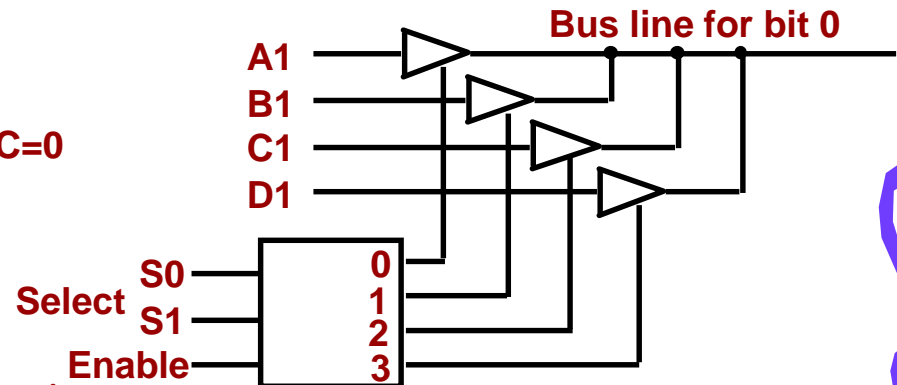
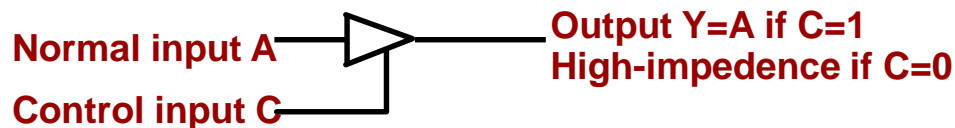
- From a register to bus: $BUS \leftarrow R$



Bus Transfer: Bus to Destination Reg



Three-State Bus Buffers



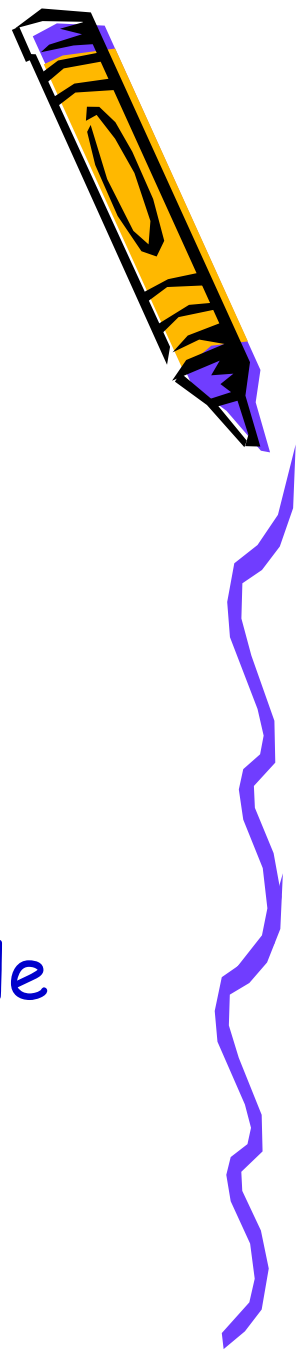
3-State Buf vs Mux implementation

Pros: area efficient & more scalable

Cons: probably more power consumption (leakage current in high-Z)



Bus vs. Point-To-Point



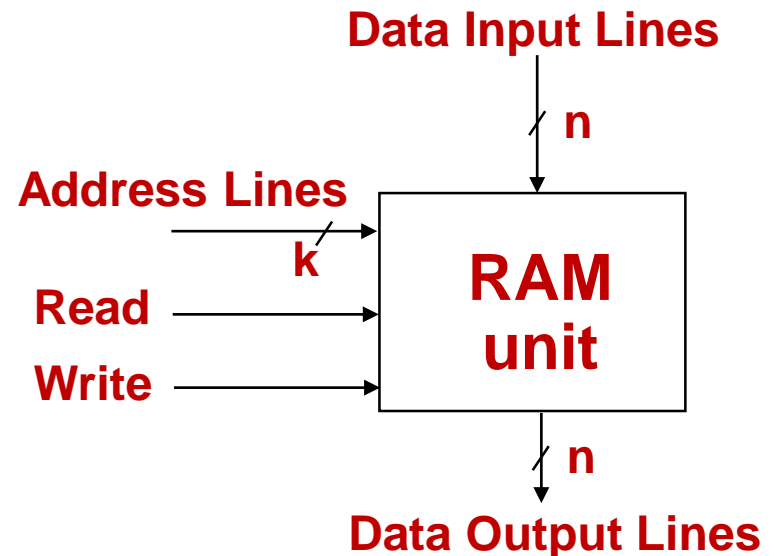
- Pros (Advantages)
 - Much less routing
 - Less area
 - Easy to scale
- Cons (Disadvantages)
 - Only one data transfer per clock cycle



Memory & Memory Transfer



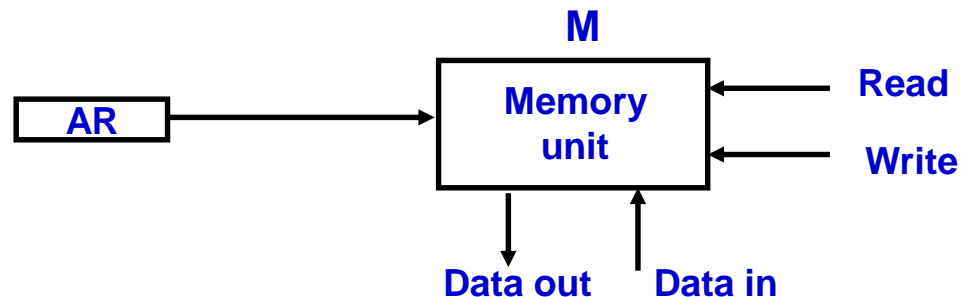
- Memory (RAM)
 - Can be thought as a sequential circuit containing some number of registers
- Assume RAM with $r = 2^k$ Words
 - n data input lines
 - n data output lines
 - k address lines
 - A read control line
 - A write control line



Memory & Memory Transfer (cont.)



- Memory Transfer
 - Memory viewed at register level as a device (M)
 - Should specify which address in memory
 - Since it contains multiple locations
- Memory Accesses
 - Provide target address in a special register
 - Memory Address Register (MAR, or AR)
 - Contents of MAR sent to memory address lines



Memory & Memory Transfer (cont.)



- Memory Read
 - To read a value from a location in memory and load it into a register
 - RTL notation looks like this:

MAR \leftarrow Address

R1 \leftarrow M[MAR]



Memory & Memory Transfer (cont.)



- Memory Write

- To write a value from a reg to a location in memory
- RTL notation looks like this:

MAR \leftarrow address

M[MAR] \leftarrow R1

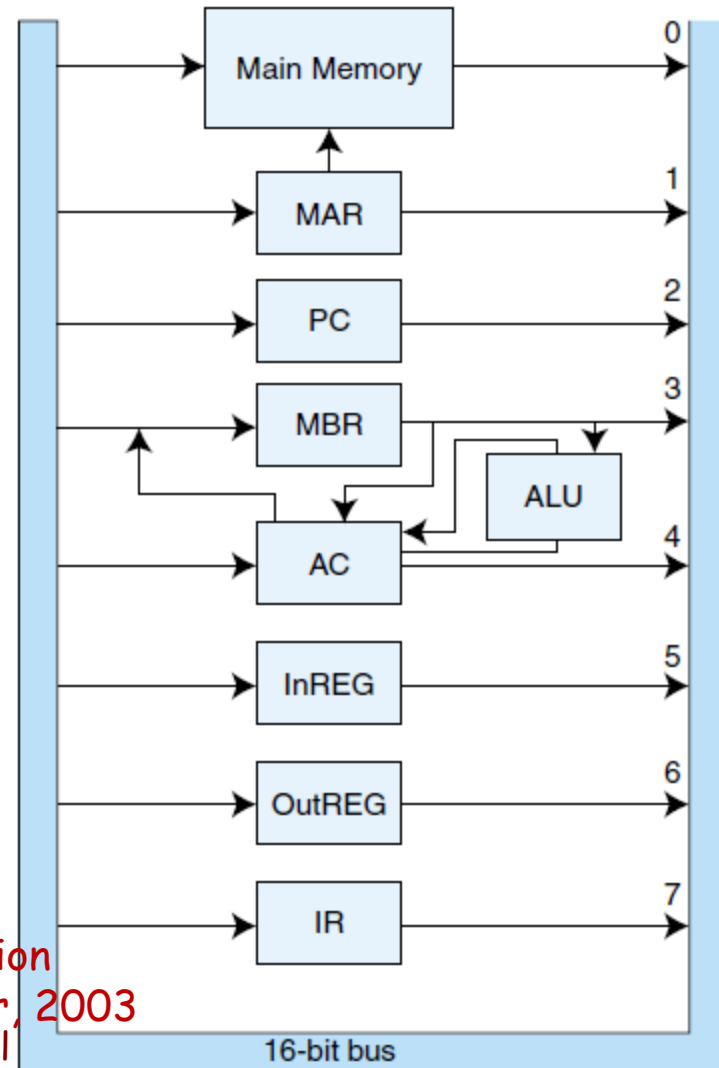


RTL Example

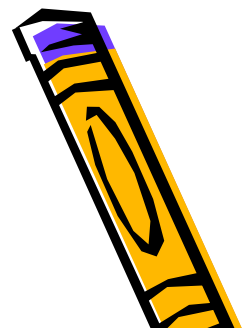


- Memory Buffer Register
 - MBR
- Accumulator
 - AC
- Program Counter
 - PC
- Memory Address Register
 - MAR
- Instruction Register
 - IR

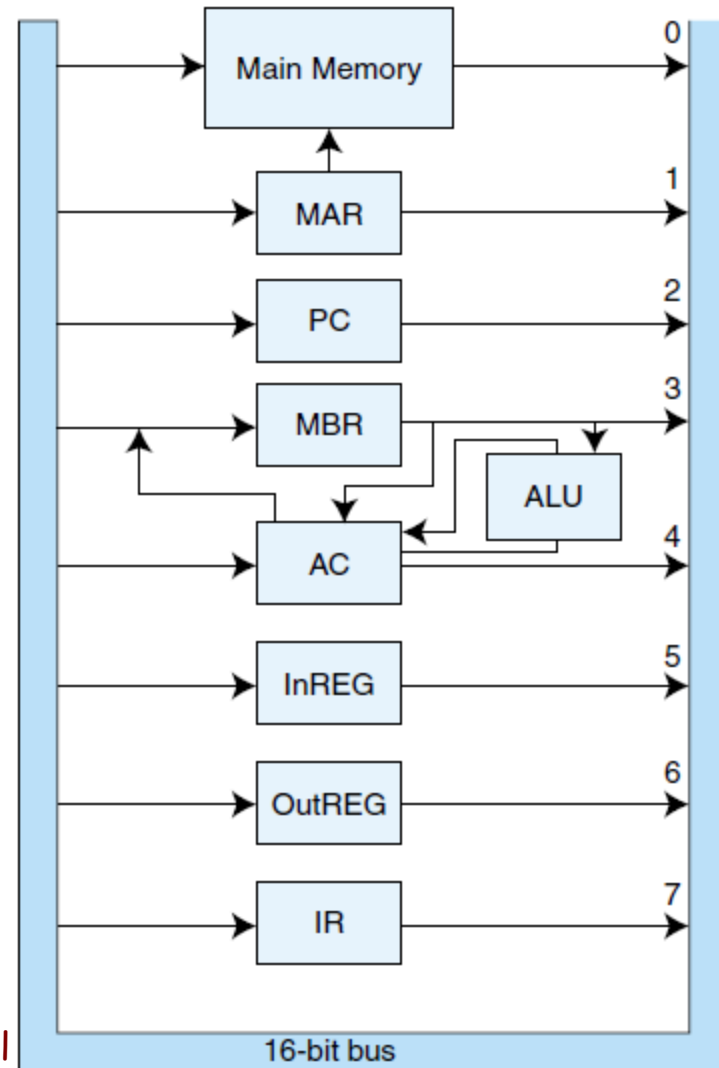
The Essentials of Computer Organization
& Architecture, by L. Null and J. Lobur, 2003
Lecture 3 Sharif University of Technol



RTL Example (cont.)

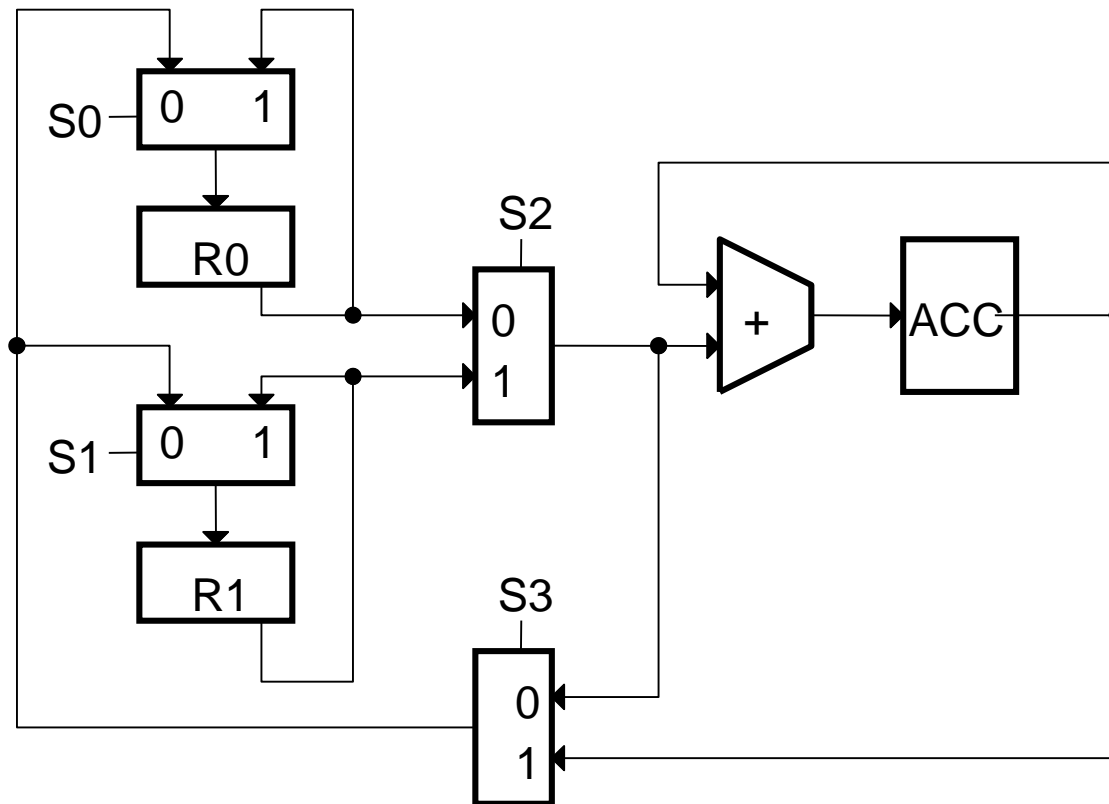


- Store X
 - $MAR \leftarrow X, MBR \leftarrow AC$
 - $M[MAR] \leftarrow MBR$
- Add X
 - $MAR \leftarrow X$
 - $MBR \leftarrow M[MAR]$
 - $AC \leftarrow AC + MBR$



Practice

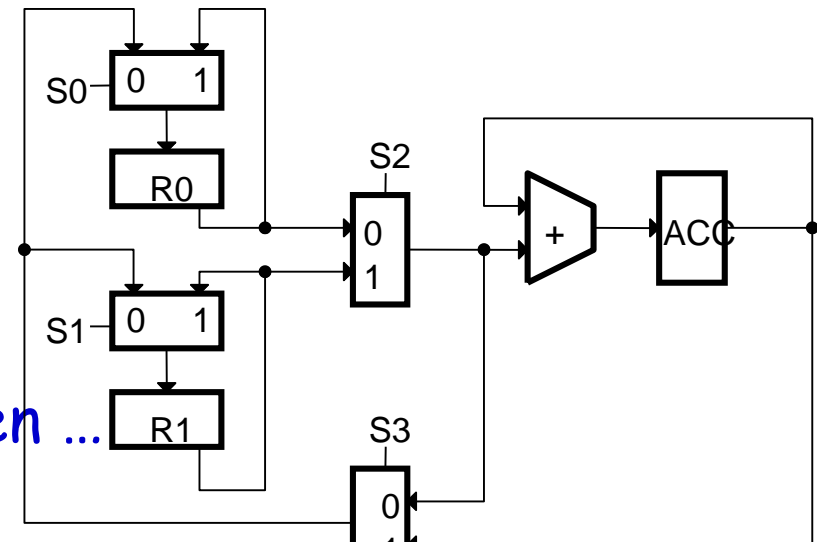
- What is RTL description of this circuit?



Practice (cont.)



- In Each Clock Cycle:
 - $\sim S3.\sim S2.\sim S1: R1 \leftarrow R0$
 - $\sim S3.S2.\sim S0: R0 \leftarrow R1$
 - $S3.\sim S2.\sim S1: ACC \leftarrow R0 + ACC$
 - $S3.S2.\sim S0: ACC \leftarrow R1 + ACC$
 - What else?
 - $R1 \leftarrow ACC$
 - ...
- Or use "if-then-else"
 - If ($S3=1$ and $S1=0$) then ...



Backup

