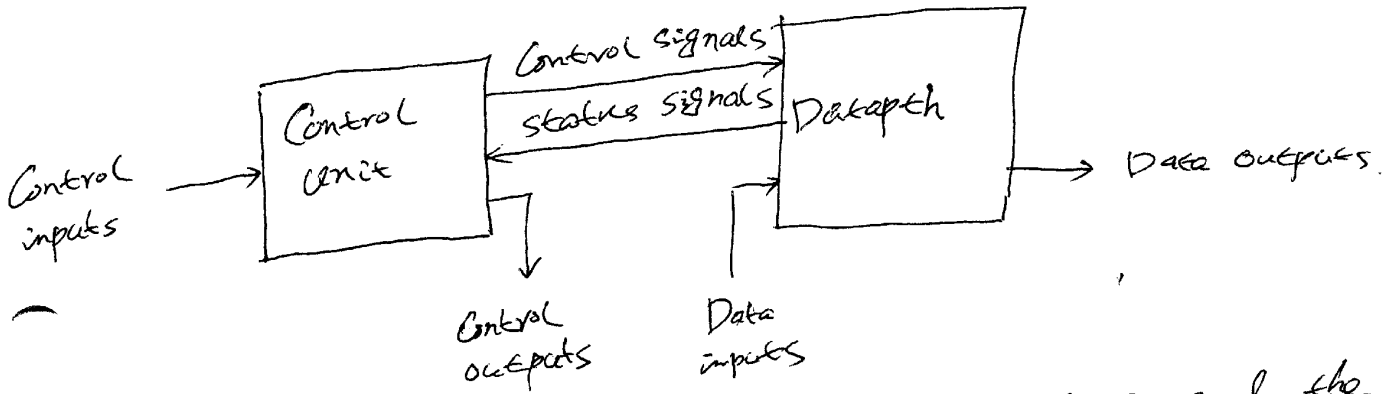


# Chapter 7 Register Transfers & Datapaths.

## 7-1. Datapaths & Operations.

In most digital system designs, we partition the system into two types of modules: a datapath, which performs data-processing operations, and a Control Unit, which determines the sequence of those operations.

Figure 7-1 Interaction between Datapath & Control Unit P. 340.



Notes:

① Datapaths are here defined by their registers and the operations that are performed on binary data stored in the registers.

② Examples of register operations are shift, count, clear, and Load.

③ The movement of the data stored in registers and the processing performed on the data are referred to as register transfer operations.

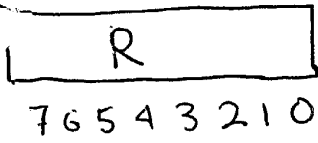
④ The elementary operations performed on the data stored in registers are called microoperations.

Examples of ~~micro~~ microoperations are loading the contents of one register into another, adding the contents of two registers, and incrementing the ~~the~~ contents of a register. → see back!

⑤ Register transfer language (RTL) is used for representing registers and specifying the operations on their contents. RTL uses a set of expressions and statements that resemble statements used in HDLs and programming language.

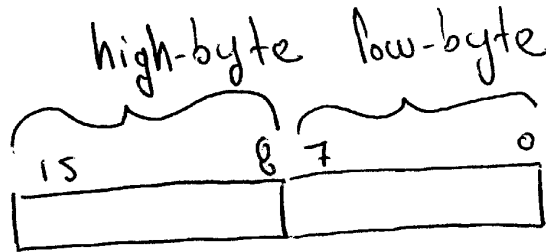
# Chapter 7: REGISTER TRANSFERS 7-1

## For 8-bits: 7.2 Register Transfer Operations AND DATAPATHS

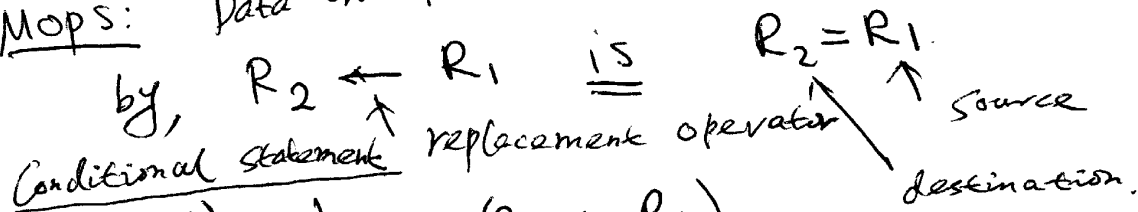


are the bits in the register,  
(little-endian notation)  
(vs. big-endian which is reversed).

## For 16-bits:



Mops: Data transfer from one register to another is designated by,



if ( $k_1 = 1$ ) then ( $R_2 \leftarrow R_1$ )

OR (a more concise way)

$k_1: R_2 \leftarrow R_1$

if ( $k_1$  is 1) then  $R_2 = R_1$ .

Executed at the same time:

$k_3: R_2 \leftarrow R_1, R_1 \leftarrow R_2$

is a swap between  $R_1$  &  $R_2$ .

"j" means simultaneous transfers.

→ See back!

Table 7-1 Basic Symbols for Register Transfers

Symbol	Description	Examples
Letters (and numerals)	Denotes a register	AR, R2, DR, IR
Parentheses	Denotes a part of a register	R2(C1), R2(7:0), AR(CL)
Arrow	Denotes transfer of data	$R1 \leftarrow R2$
Comma	Separates simultaneous transfers	$R1 \leftarrow R2, R2 \leftarrow R1$
Square brackets	Specifies an address for memory	$DR \leftarrow M[AR]$ <div style="margin-left: 150px;"> <math>\uparrow</math>                      a memory word -  <del>denotes</del> </div>

2's complement:

$$R_0 \leftarrow R_1 + \overline{R_2} + 1 \quad \text{is} \quad R_0 = R_1 - R_2$$

→ See back for table 7-3.

this means compute 1's complement

$\overline{R_2} + 1$  for 2's complement of  $R_2$ .

addition:  $R_1 = R_1 + R_2$   
 subtraction:  $R_1 = R_1 - R_2$

both X, K, must be true for execution.

Logic M-operations:

$V = OR, \quad \wedge = AND$  → see back!

Consider:  
 1. set 0th & 3rd bits of  $R_1$  to 1.  
 (leave other bits "as they are").

solution:  $R_1 \leftarrow R_1 \vee (1001)_2$

$\begin{matrix} \uparrow & \uparrow & \uparrow & \uparrow \\ 3rd & 2nd & 1st & 0th \end{matrix}$

2. Mask out upper 8 bits of 16-bit register  $R_1$ , so that the upper 8-bits are all zero.

$$R_1 \leftarrow R_1 \wedge (0000\ 0000\ 1111\ 1111)_2$$

$\underbrace{\hspace{10em}}_{\text{all 0s}} \quad \underbrace{\hspace{10em}}_{\text{all 1s}}$

16-bits.

Table 7-3

Arithmetic Microoperations

Symbolic designation	Description
$R_0 \leftarrow R_1 + R_2$	Contents of $R_1$ plus $R_2$ transferred to $R_0$
$R_2 \leftarrow \bar{R}_2$	Complement of $R_2$ (1's Complement)
$R_2 \leftarrow \bar{R}_2 + 1$	2's Complement of $R_2$
$R_0 \leftarrow R_1 + \bar{R}_2 + 1$	Subtraction $R_0 \leftarrow R_1 - R_2$
$R_1 \leftarrow R_1 + 1$	Increment the contents of $R_1$ (Count up)
$R_1 \leftarrow R_1 - 1$	Decrement the contents of $R_1$ (Count down)

Table 7-4

Logic Microoperations

Symbolic designation	Description
$R_0 \leftarrow \bar{R}_1$	Logic bitwise NOT (1's Complement)
$R_0 \leftarrow R_1 \wedge R_2$	" " AND (cleans bits)
$R_0 \leftarrow R_1 \vee R_2$	" " OR (sets bits)
$R_0 \leftarrow R_1 \oplus R_2$	" " XOR (complements bits)

3. Set lower bits of CR to 101:

$$CR \leftarrow CR \vee (101)_2 \quad (\vee \text{ logical OR})$$

$$CR \leftarrow CR \wedge (1111 \ 1111 \ 1111 \ 1101)_2 \quad (\wedge \text{ logical AND})$$

↑ assuming 16-bit Control register.

4. Flip bits 1 & 3 of CR:

$$CR \leftarrow CR \oplus (1010)_2$$

↑    ↑  
3rd 1st

(K1+K2): R1 ← R2 + R3,   
 ↑ + add  
 OR                      R4 ← R5 ∨ R6.  
 operation                      ↓ OR  
 in a control condition.  
 K1, K2 true or false,  
 Boolean Variable

Shifts (sl, sr).

→ see back!

Consider:

1. Multiply R<sub>1</sub> by 2 w/out multiplying.

$$R_1 \leftarrow sl \ R_1$$

2. Multiply R<sub>1</sub> by 5 w/out multiplying:

$$5 = (101)_2$$

1. 2 left shifts    ↑    ↑  
 + 0.1 left shifts    + 1. 0 left shifts.

$$\left\{ \begin{array}{l} R_2 \leftarrow R_1 \quad (0 \text{ shift}) \\ R_1 \leftarrow sl \ R_1 \\ R_1 \leftarrow sl \ R_1 \\ R_1 \leftarrow R_1 + R_2 \end{array} \right.$$

Table 7-5.

Examples of Shifts.

Eight bit examples.

Type	Symbolic designation	Source R <sub>2</sub>	After shift: Destination R <sub>1</sub>
Shift left	$R_1 \leftarrow SL R_2$	10011110	00111100
Shift right	$R_1 \leftarrow SR R_2$	11100101	01110010

3. Divide integer  $R_1$  by 2, and truncate the result.

If  $R_1 \geq 0$ :

$$R_1 \leftarrow sr R_1$$

If  $R_1 < 0$ :

2's Complement.

$$R_1 \leftarrow sr R_1$$

$$R_1 \leftarrow R_1 \vee (1000\ 0000\ 0000\ 0000)_2$$

↑  
sign-bit

division →

Eg:  $\left\{ \begin{array}{l} 0 \rightarrow (1000\ 0000)_2 -128 \\ \text{shift to the right is:} \end{array} \right.$

$$\begin{array}{r} (0100\ 0000)_2 \\ 1100\ 0000 \\ \hline -64 \end{array}$$

Multiplication →

$\left\{ \begin{array}{l} (0000\ 0001)_2 \leftarrow 0 \\ \text{shift to the left is:} \end{array} \right.$

$$\begin{array}{r} 1000\ 0001 \\ 0100\ 0001 \end{array}$$

$$\begin{array}{r} (0000\ 0010)_2 \\ 1000\ 0010 \\ 0111\ 1101 + \\ \hline 0111\ 1110 \end{array}$$

Figure 7-5:

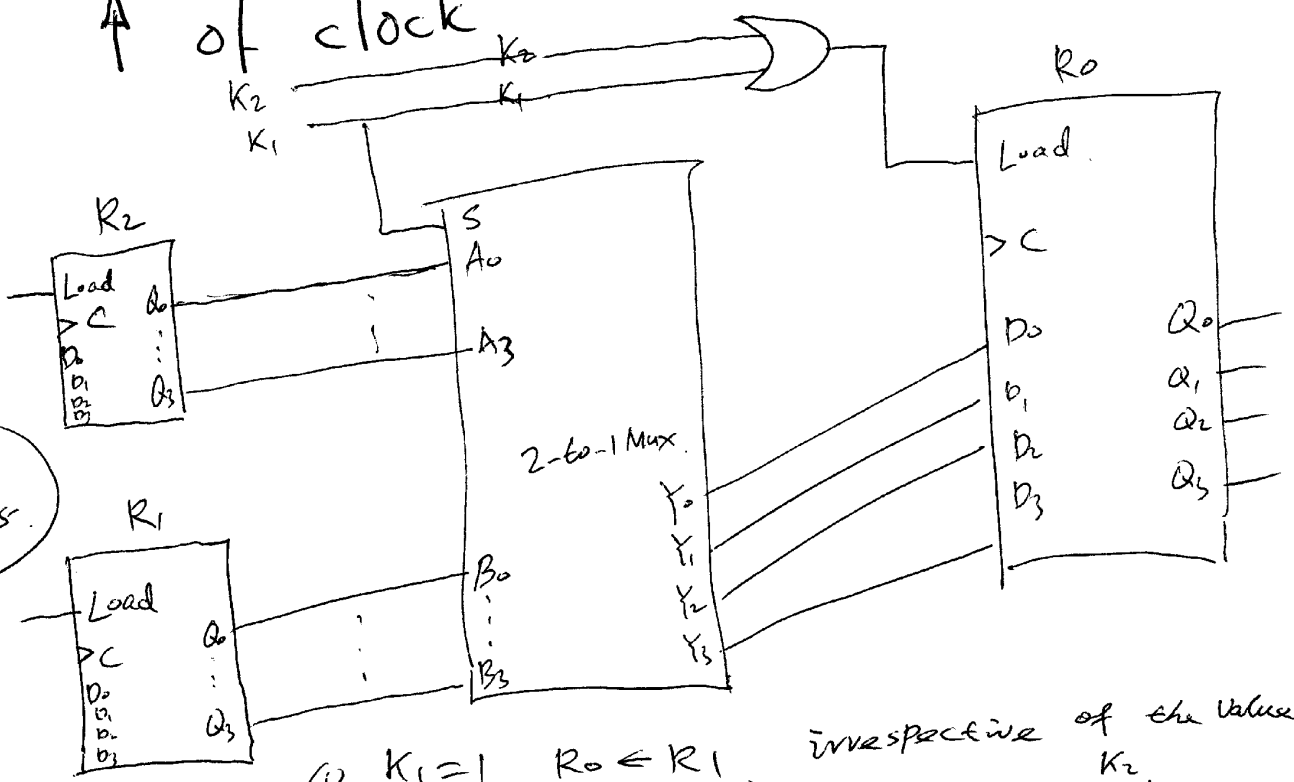
$\underline{\text{If}} (k_1 = 1)$   
 $\underline{\text{then}} R_0 \leftarrow R_1$   
 $\underline{\text{else if}} (k_2 = 1)$   
 $\underline{\text{then}} R_0 \leftarrow R_2$

is equivalent to:

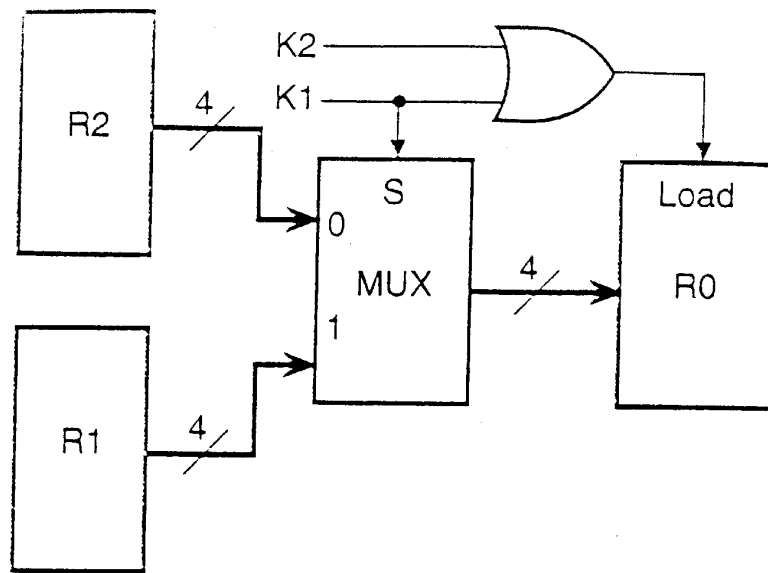
$k_1: R_0 \leftarrow R_1, \bar{k}_1 k_2: R_0 \leftarrow R_2$

on  $\uparrow$  of clock

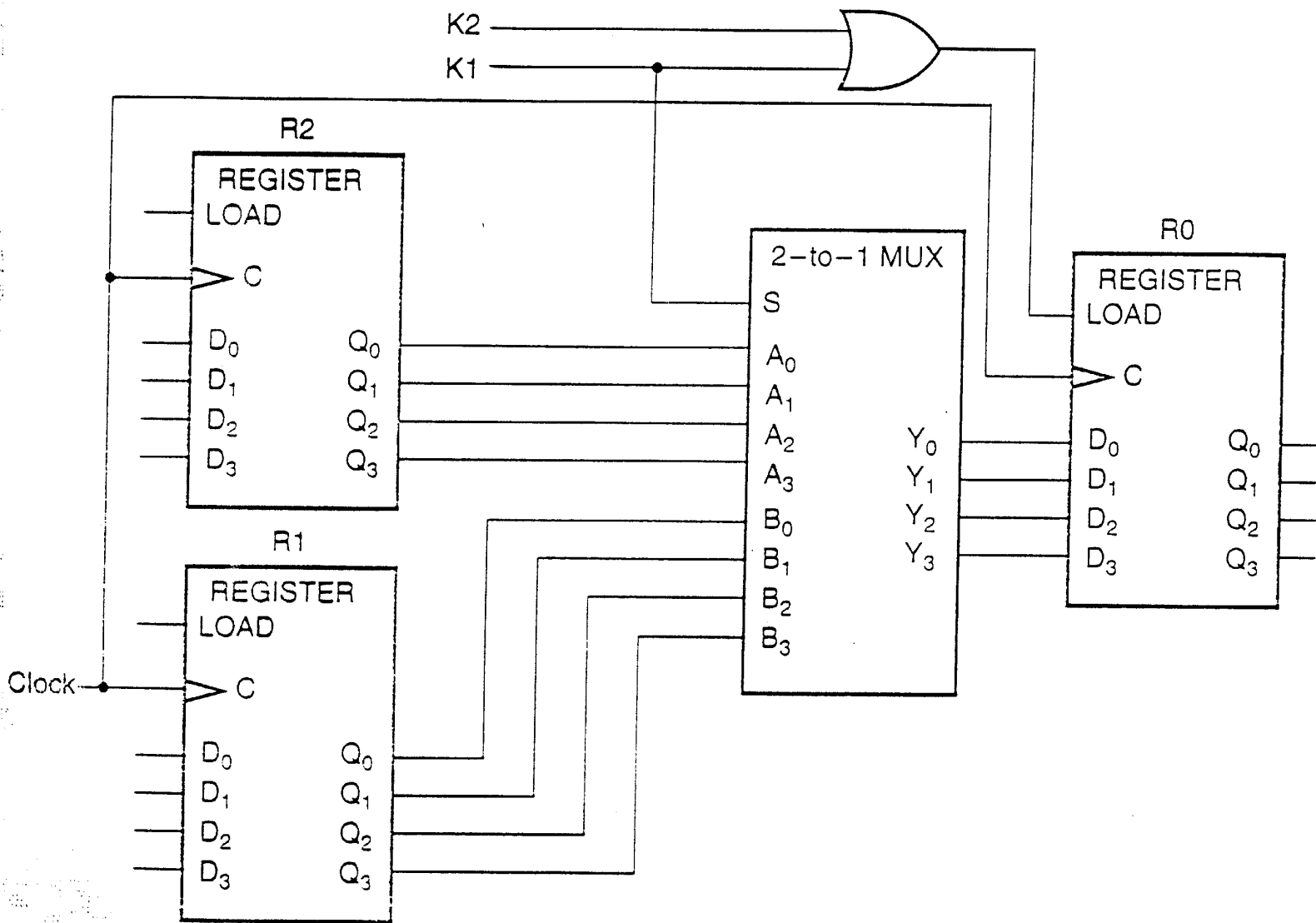
page 351  
Figure 7-5.



- ①  $k_1 = 1, R_0 \leftarrow R_1$ , irrespective of the value of  $k_2$ .
- ②  $k_1 = 0, k_2 = 1, R_0 \leftarrow R_2$ .
- ③  $k_1 = k_2 = 0, \text{Load} = 0$  of  $R_0$ .



(a) Block diagram



(b) Detailed logic

□ FIGURE 7-5

Use of Multiplexers to Select between Two Registers

Memory Transfer

Read:  $DR \leftarrow M[AR]$

Write:  $M[AR] \leftarrow DR$

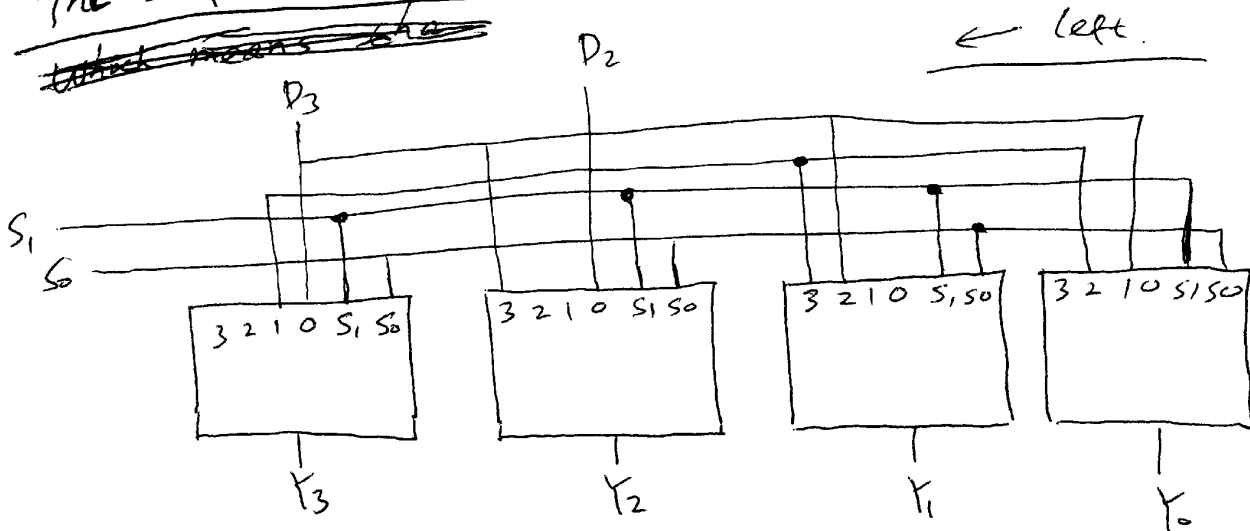
7-8 The shifter.

Barrel shifter.

In some datapath applications, the data must be shifted more than one bit position in a single clock cycle.

A barrel shifter is one form of combinational circuit that shifts or rotates the input data bits by the number of bit positions specified by a binary value on a set of selection lines.

The shift we consider here is a rotation to the left.



P367  
Fig 7-17

Table 7-9  
Function Table for 4-bit barrel shifter.

select		output				operation
$S_1$	$S_0$	$Y_3$	$Y_2$	$Y_1$	$Y_0$	
0	0	$D_3$	$D_2$	$D_1$	$D_0$	No rotation.
0	1	$D_2$	$D_1$	$D_0$	$D_3$	rotate one position.
1	0	$D_1$	$D_0$	$D_3$	$D_2$	rotate two positions.
1	1	$D_0$	$D_3$	$D_2$	$D_1$	rotate three positions.