Processor and Computer (Supplementary Materials)

John SUM
Institute of Technology Management
National Chung Hsing University
Taichung, ROC

October 22, 2021

Contents

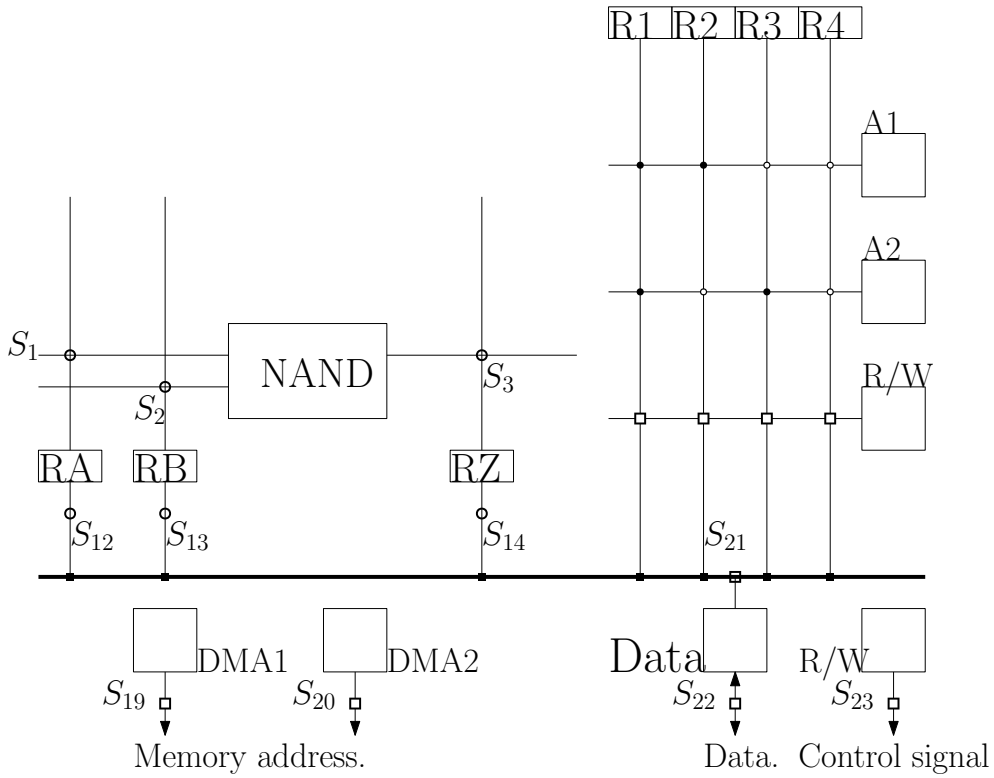

Figure 1: A single-logic-gate processor.

1 Single-NAND-Gate Processor

Figure 1 shows an artificial processor with a single NAND gate and a few registers. The registers, $R1$ to $R4$, are connected to the data line via three different switches. The empty (resp. solid) black circle switch is 'ON' if the control signal is '1' (resp. '0'). These switches are connected to a two-signal generator for $A1$ and $A2$. The switches connecting the R/W are two-way switches. The signals sending to $A1$, $A2$ and R/W together with the corresponding actions are depicted in the following table.

$A1$	$A2$	R/W	Action
0	0	01	Read data from $R1$
0	1	01	Read data from $R2$
1	0	01	Read data from $R3$
1	1	01	Read data from $R4$
0	0	10	Write data to $R1$
0	1	10	Write data to $R2$
1	0	10	Write data to $R3$
1	1	10	Write data to $R4$
x	x	00	Disconnection

2 Simple Logical Operations

To perform simple logical operations, more than one micro-instruction are needed. The micro-instructions for a few simple logical operations are shown in Figure 2. To describe the steps clearly, let me introduction a notation $\odot$ to denote the NAND operation.

2.1 NOT RA

For the micro-instructions for 'NOT RA', there are two steps. Before execution, the data is already available in RA.

S1. $RB = RA$.

S2. $RZ = RA \odot RB$.

2.2 AND RA RB

For the micro-instructions for 'AND RA RB', there are three steps. Before execution, the data are already available in RA and RB.

S1. $RZ = RA \odot RB$.

S2. $RA = RZ$. $RB = RZ$.

S3. $RZ = RA \odot RB$.

2.3 OR RA RB

For the micro-instructions for 'OR RA RB', there are nine steps. Registers $R1$ and $R2$ are needed. Before execution, the data are already available in RA and RB.

S1. $R1 = RB$.

S2. $RB = RA$.

S3. $RZ = RA \odot RB$.

S4. $R2 = RZ$.

S5. $RA = R1$. $RB = R1$.

S6. $RZ = RA \odot RB$.

S7. $RB = RZ$.

S8. $RA = R2$.

S9. $RZ = RA \odot RB$.

2.4 NOR RA RB

For the micro-instructions for 'NOR RA RB', there are eleven steps. Registers $R1$ and $R2$ are needed. The first nine steps are identical to the steps for 'OR RA RB'. Before execution, the data are already available in RA and RB.

S1. $R1 = RB$.

S2. $RB = RA$.

S3. $RZ = RA \odot RB$.

S4. $R2 = RZ$.

S5. $RA = R1$. $RB = R1$.

S6. $RZ = RA \odot RB$.

S7. $RB = RZ$.

S8. $RA = R2$.

S9. $RZ = RA \odot RB$.

S10. $RA = RZ$. $RB = RZ$.

S11. $RZ = RA \odot RB$.

Instructions	S_1	S_2	S_3	S_{12}	S_{13}	S_{14}	A_1	A_2	R/W
NOT RA	0	0	0	01	10	00	0	0	00
	1	1	1	00	00	00	0	0	00
AND RA RB	1	1	1	00	00	00	0	0	00
	0	0	0	10	10	01	0	0	00
	1	1	1	00	00	00	0	0	00
OR RA RB	0	0	0	00	01	00	0	0	10
	0	0	0	01	10	00	0	0	00
	1	1	1	00	00	00	0	0	00
	0	0	0	00	00	01	0	1	10
	0	0	0	10	10	00	0	0	01
	1	1	1	00	00	00	0	0	00
	0	0	0	00	10	01	0	0	00
	0	0	0	10	00	00	0	1	01
	1	1	1	00	00	00	0	0	00
NOR RA RB	0	0	0	00	01	00	0	0	10
	0	0	0	01	10	00	0	0	00
	1	1	1	00	00	00	0	0	00
	0	0	0	00	00	01	0	1	10
	0	0	0	10	10	00	0	0	01
	1	1	1	00	00	00	0	0	00
	0	0	0	00	10	01	0	0	00
	0	0	0	10	00	00	0	1	01
	1	1	1	00	00	00	0	0	00
	0	0	0	10	10	01	0	0	00
	1	1	1	00	00	00	0	0	00

Figure 2: Sample micro-instructions for the single-logic-gate processor as shown in Figure 1.

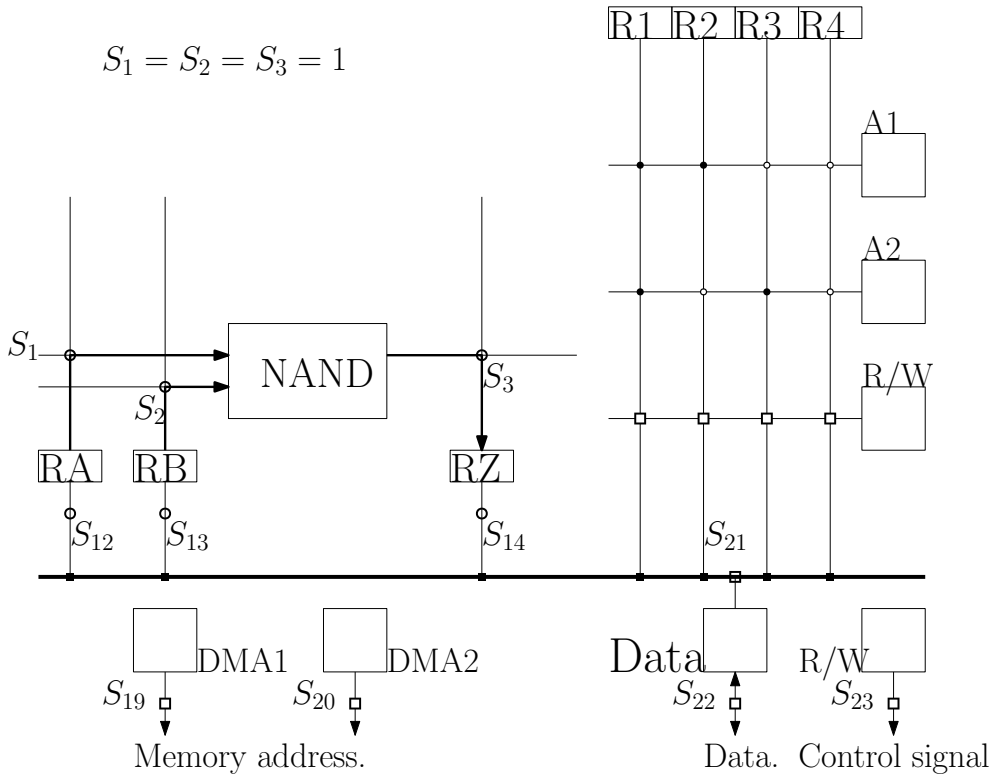

Figure 3: Implementation of the first step, $RZ = RA \odot RB$, for the instruction $AND\ RA\ RB$. The thick arrows indicate the flow of electrical signals.

For illustration, the flow of the electrical signals during the implementation of the instruction $AND\ RA\ RB$ are shown in Figure 3, Figure 4 and Figure 5.

3 Micro-Instruction Design

For the single-NAND-gate processor as shown in Figure 1, the number of micro-instructions for the realization of a logical operation is different from one to another. Consider that the completion time of one micro-instruction takes one clock cycle. The time taken for the processor to perform a logical operation is equal to the number of micro-instructions to be executed.

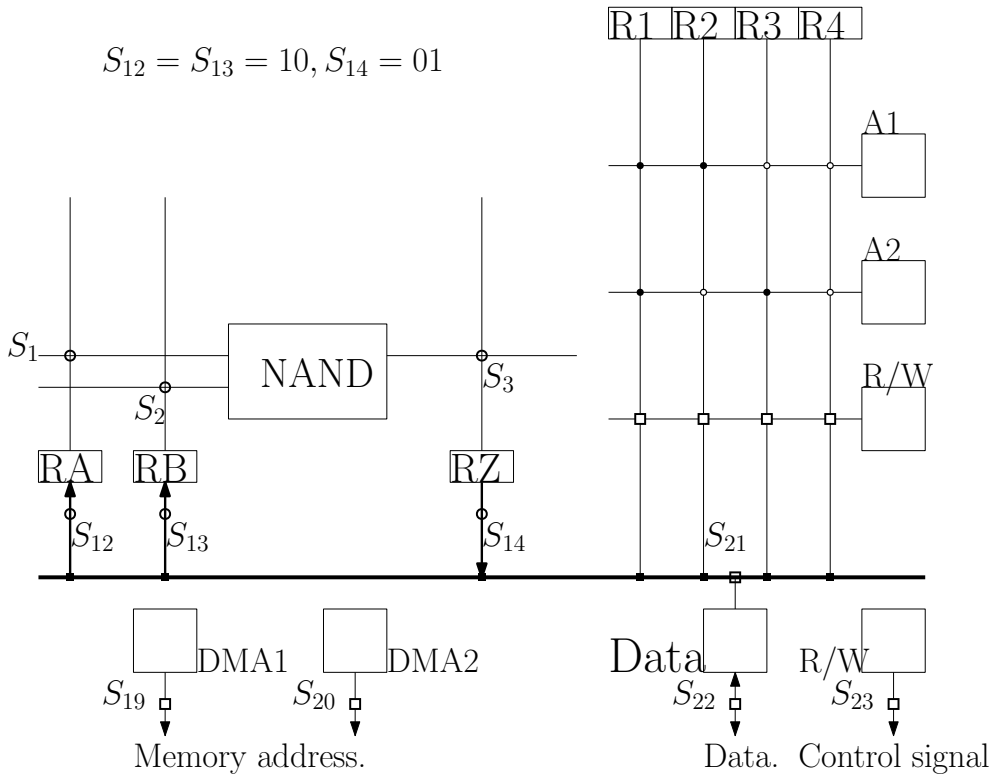

Figure 4: Implementation of the second step, $RA = RZ$ and $RB = RZ$, for the instruction $AND\ RA\ RB$. The thick arrows indicate the flow of electrical signals.

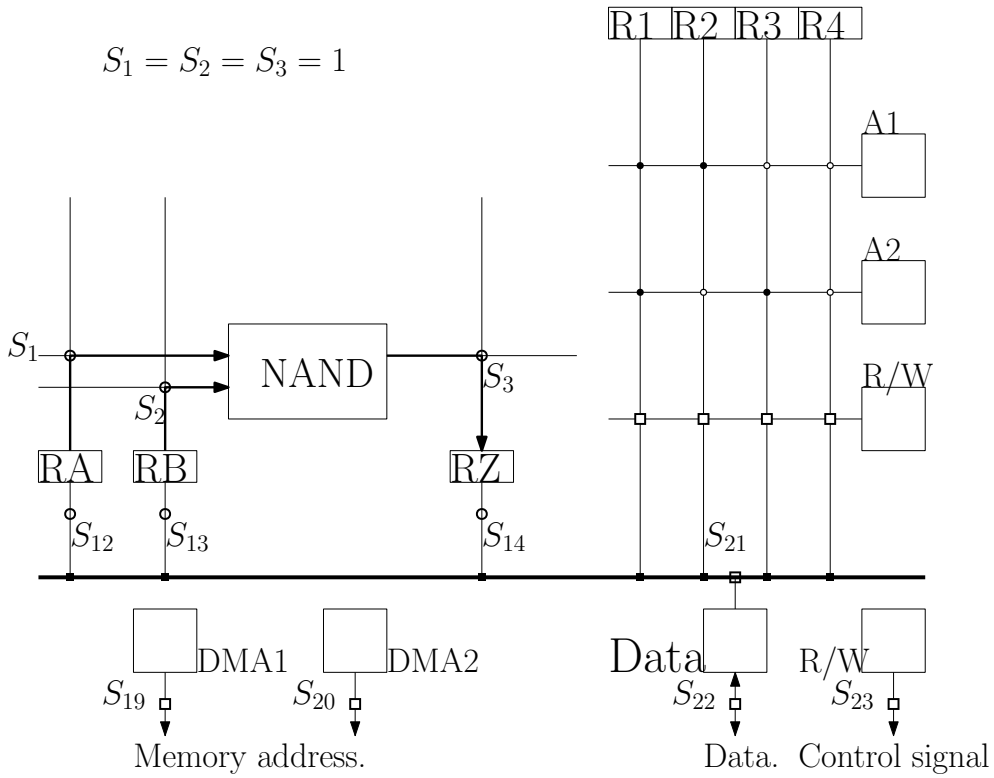

Figure 5: Implementation of the third step, $RZ = RA \odot RB$, for the instruction *AND RA RB*. The thick arrows indicate the flow of electrical signals.

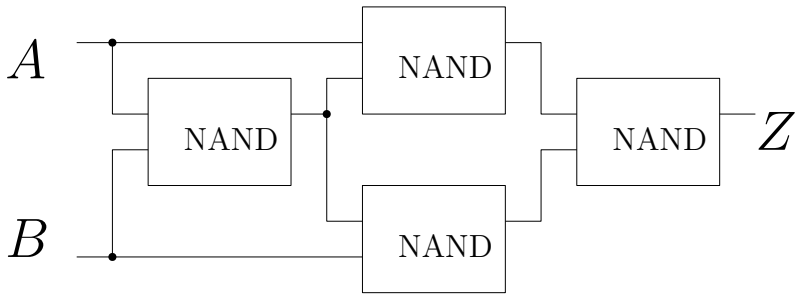

Figure 6: Implementation of an XOR gate using NAND gates.

Logical Operation	NOT	AND	OR	NOR
No. of Micro-Instructions	2	3	9	11
No. of Clock Cycles	2	3	9	11

Therefore, a poor design on the micro-instructions might end up spend more time to complete the jobs to be done for an instruction.

3.1 XOR RA RB

To illustrate the important of micro-instruction design, let us have an example about the logical operation XOR. To realize the instruction 'XOR RA RB', one would need to have the digital logic circuit for the implementation of an XOR gate using NAND gates. Its logical equation is given as follows :

$$A \oplus B = (A \odot (A \odot B)) \odot (B \odot (A \odot B)) .$$

It is to recall that the notation $\oplus$ denotes the XOR operation and the notation $\odot$ denotes the NAND operation.

3.2 Algorithm I for XOR RA RB

With reference to the digital logic circuit in Figure 6, the following algorithm can be applied.

S1. $R1 = RA$.

S2. $R2 = RB$.

S3. $RZ = RA \odot RB$.

S4. $RB = RZ$.

S5. $R3 = RZ$.

S6. $RZ = RA \odot RB$.

S7. $R4 = RZ$.

S8. $RA = R2$.

S9. $RZ = RA \odot RB$.

S10. $RA = R4$.

S11. $RB = RZ$.

S12. $RZ = RA \odot RB$.

There are twelve micro-instructions. Four additional registers are needed as the working memory space for the algorithm. The contents of the registers after an instruction has been executed are shown in Figure 7. The micro-instructions for Algorithm I are listed in Figure 8.

3.3 Algorithm II for XOR RA RB

Pretty clear, Algorithm I presented in the previous section is not the only algorithm which can implement the XOR operation. Besides, Algorithm I might not be the optimal algorithm for the realization of the XOR operation. From the algorithm design point of view, if possible, one should design the optimal solution for a problem.

The optimal solution is the algorithm which has the minimum number of micro-instructions and requires the minimum number of registers for the implementation of a logical operation.

One might have been aware that the fourth and the firth micro-instructions in the Algorithm I can be executed simultaneously. Thus, Algorithm I is not the optimal solution. Following the digital circuit as shown in Figure 6, we can have an alternative algorithm.

S1. $R1 = RB$.

S2. $RZ = RA \odot RB$.

Step	RA	RB	RZ	$R1$	$R2$	$R3$	$R4$
0	A	B	-	-	-	-	-
1	A	B	-	A	-	-	-
2	A	B	-	A	B	-	-
3	A	B	$A \odot B$	A	B	-	-
4	A	$A \odot B$	$A \odot B$	A	B	-	-
5	A	$A \odot B$	$A \odot B$	A	B	$A \odot B$	-
6	A	$A \odot B$	$A \odot (A \odot B)$	A	B	$A \odot B$	-
7	A	$A \odot B$	$A \odot (A \odot B)$	A	B	$A \odot B$	$A \odot (A \odot B)$
8	B	$A \odot B$	$A \odot (A \odot B)$	A	B	$A \odot B$	$A \odot (A \odot B)$
9	B	$A \odot B$	$B \odot (A \odot B)$	A	B	$A \odot B$	$A \odot (A \odot B)$
10	$A \odot (A \odot B)$	$A \odot B$	$B \odot (A \odot B)$	A	B	$A \odot B$	$A \odot (A \odot B)$
11	$A \odot (A \odot B)$	$B \odot (A \odot B)$	$B \odot (A \odot B)$	A	B	$A \odot B$	$A \odot (A \odot B)$
12	$A \odot (A \odot B)$	$B \odot (A \odot B)$	$(A \odot (A \odot B)) \odot (B \odot (A \odot B))$	A	B	$A \odot B$	$A \odot (A \odot B)$

Figure 7: The contents in the registers after a micro-instruction in the Algorithm I has been executed. The number of registers, apart from RA, RB and RZ, required for the algorithm is four.

Instructions	S_1	S_2	S_3	S_{12}	S_{13}	S_{14}	A_1	A_2	R/W
XOR RA RB	0	0	0	01	00	00	0	0	10
	0	0	0	00	01	00	0	1	10
	1	1	1	00	00	00	0	0	00
	0	0	0	00	10	01	0	0	00
	0	0	0	00	00	01	1	0	10
	1	1	1	00	00	00	0	0	00
	0	0	0	00	00	01	1	1	10
	0	0	0	10	00	00	0	1	01
	1	1	1	00	00	00	0	0	00
	0	0	0	10	00	00	1	1	01
	0	0	0	00	10	01	0	0	00
	1	1	1	00	00	00	0	0	00

Figure 8: Micro-instructions for the Algorithm I.

S3. $R2 = RZ$. $RB = RZ$.

S4. $RZ = RA \odot RB$.

S5. $R3 = RZ$.

S6. $RA = R2$.

S7. $RB = R1$.

S8. $RZ = RA \odot RB$.

S9. $RA = R3$.

S10. $RB = RZ$.

S11. $RZ = RA \odot RB$.

There are eleven micro-instructions. Three additional registers are needed as the working memory space for the algorithm. The contents of the registers after an instruction has been executed are shown in Figure 9. The micro-instructions for Algorithm II are listed in Figure 10.

Step	RA	RB	RZ	$R1$	$R2$	$R3$	$R4$
0	A	B	-	-	-	-	-
1	A	B	-	B	-	-	-
2	A	B	$A \odot B$	B	-	-	-
3	A	$A \odot B$	$A \odot B$	B	$A \odot B$	-	-
4	A	$A \odot B$	$A \odot (A \odot B)$	B	$A \odot B$	-	-
5	A	$A \odot B$	$A \odot (A \odot B)$	B	$A \odot B$	$A \odot (A \odot B)$	-
6	$A \odot B$	$A \odot B$	$A \odot (A \odot B)$	B	$A \odot B$	$A \odot (A \odot B)$	-
7	$A \odot B$	B	$A \odot (A \odot B)$	B	$A \odot B$	$A \odot (A \odot B)$	-
8	$A \odot B$	B	$B \odot (A \odot B)$	B	$A \odot B$	$A \odot (A \odot B)$	-
9	$A \odot (A \odot B)$	B	$B \odot (A \odot B)$	B	$A \odot B$	$A \odot (A \odot B)$	-
10	$A \odot (A \odot B)$	$B \odot (A \odot B)$	$B \odot (A \odot B)$	B	$A \odot B$	$A \odot (A \odot B)$	-
11	$A \odot (A \odot B)$	$B \odot (A \odot B)$	$(A \odot (A \odot B)) \odot (B \odot (A \odot B))$	B	$A \odot B$	$A \odot (A \odot B)$	-

Figure 9: The contents in the registers after a micro-instruction in the Algorithm II has been executed. The number of registers, apart from RA, RB and RZ, required for the algorithm is four.

Instructions	S_1	S_2	S_3	S_{12}	S_{13}	S_{14}	A_1	A_2	R/W
XOR RA RB	0	0	0	00	01	00	0	0	10
	1	1	1	00	00	00	0	0	00
	0	0	0	00	10	01	0	1	10
	1	1	1	00	00	00	0	0	00
	0	0	0	00	00	01	1	0	10
	0	0	0	10	00	00	0	1	01
	0	0	0	00	10	00	0	0	01
	1	1	1	00	00	00	0	0	00
	0	0	0	10	00	00	1	0	01
	0	0	0	00	10	01	0	0	00
	1	1	1	00	00	00	0	0	00

Figure 10: Micro-instructions for the Algorithm II.

3.4 Complexity

In the area of computer science, the performance of an algorithm is determined by the number of steps (computational complexity) and the amount of memory space (memory complexity) required for the completion of that algorithm.

Algorithm	No. of M.I.	No. of Registers
NOT RA	2	0
AND RA RB	3	0
OR RA RB	9	2
NOR RA RB	11	2
XOR RA RB (Algo. I)	12	4
XOR RA RB (Algo. II)	11	3

From the above table, based on the single-NAND-gate processor, the computational complexity (resp. memory complexity) required for the implementation of the Algorithm II is smaller than the computational complexity (resp. memory complexity) required for the implementation of the Algorithm I.

Therefore, the performance of processor is not just determined by its clock speed. It is also determined by the design of the micro-instructions.

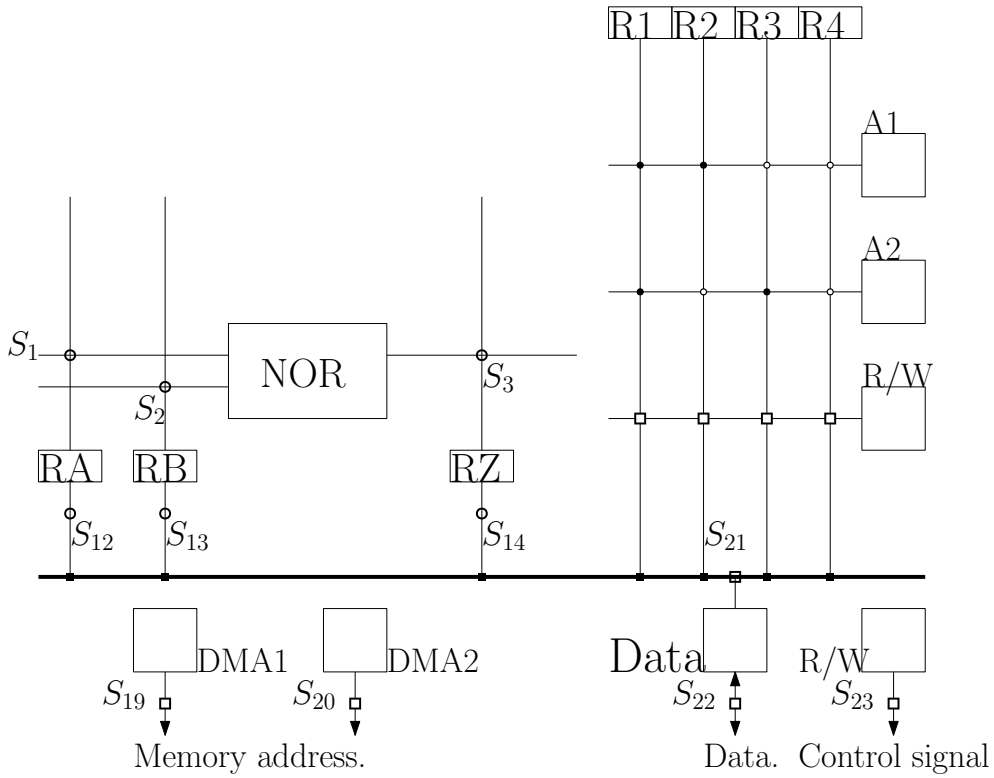

Figure 11: A single-NOR-gate processor.

The design of a micro-program is determined by the architecture of the processor. Poor micro-program design or poor processor architecture design would degrade the performance of a processor.

4 Exercises

Figure 11 shows a processor with single NOR gate. Its architecture is the same as the one shown in Figure 1 except that the NAND gate is replaced by a NOR gate. The truth table of a NOR gate is depicted in the following table.

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

Truth Table of a NOR Gate.

You would need to design the micro-instructions for the implementation of the following logical operations.

(a) NOT RA.

(b) AND RA RB.

(c) OR RA RB.

(d) NAND RA RB.

(e) XOR RA RB.

As a reference, the digital logic circuits for the implementation of AND gate and NAND gate using NOR gates are shown in Figure 12. For the implementation of the XOR gate, some digital logic circuits could be found on the Wikipedia[1].

Like the examples presented in the previous sections. You need to count for each micro-program (i) its number of micro-instructions and (ii) the number of registers being used.

[1] `https://en.wikipedia.org/wiki/NOR_logic`.

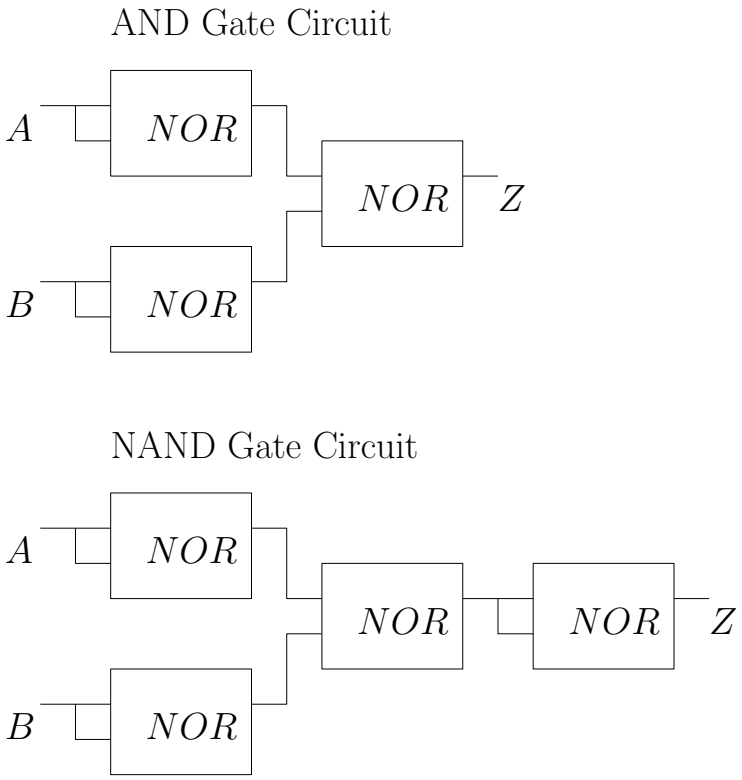

Figure 12: The digital logic circuits for the implementation of AND gate and NAND gate using NOR gates only.