Dela Rosa. Pamella Jei S.
2020-06676

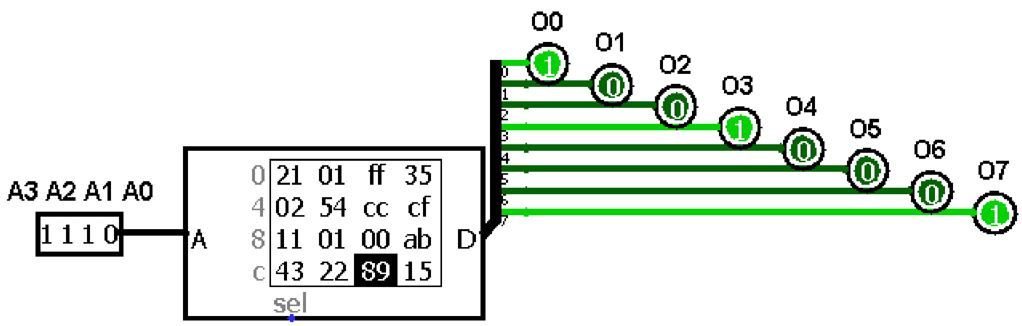
CS 20 Laboratory 10: Read Only Memory

1. (4 pts) Basics

   a. (2pts) For items 10-13 of Section 2.1, answer the question, include pictures of the circuit with the inputs and LED outputs (we expect 4), and attach the Logisim file (cs20lab10_1a.circ) of the circuit. Make sure that the circuit is well labelled on which is A4, A3, etc. and 07, 06, etc.

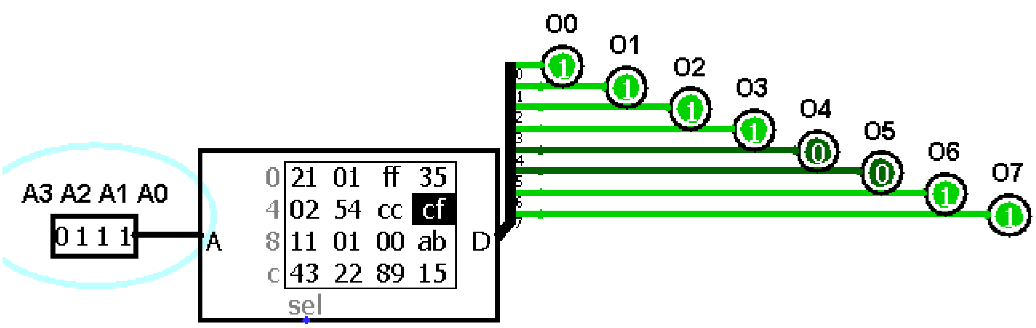
10) To access the cell containing the value 89, what should the input pins be? What pattern did you observe in the output LEDs?

The input pins should be 1110, while the output LEDs are 10001001 (with O7 as the MSB and O0 as the LSB) which is the binary equivalent of 89_{16}.



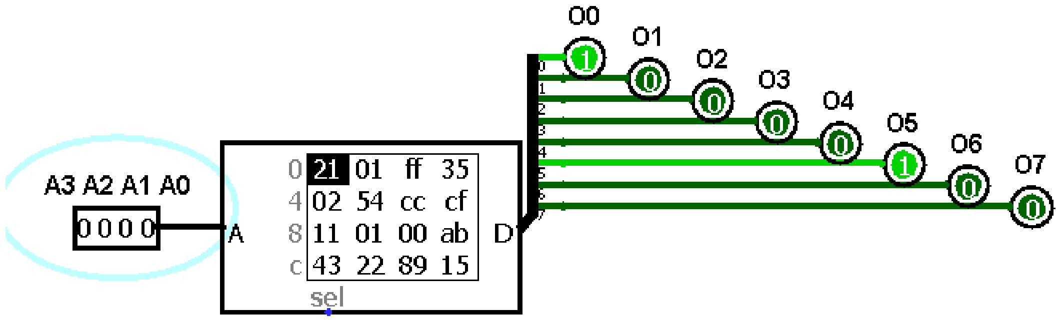
11) To access the cell containing the value cf, what should the input pins be? What pattern did you observe in the output LEDs?

The input pins should be 0111, while the output LEDs are 11001111 (with O7 as the MSB and O0 as the LSB) which is the binary equivalent of cf_{16}.



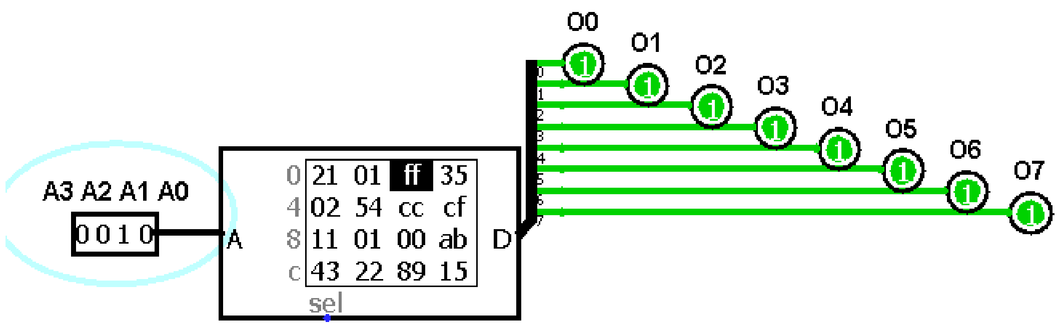
12) To access the cell containing the value 21, what should the input pins be? What pattern did you observe in the output LEDs?

The input pins should be 0000, while the output LEDs are 00100001 (with O7 as the MSB and O0 as the LSB) which is the binary equivalent of 21_{16}.



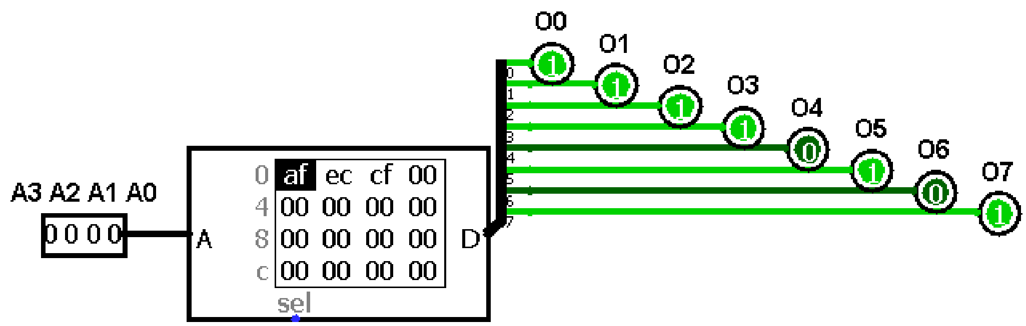
13) To access the cell containing the value ff, what should the input pins be? What pattern did you observe in the output LEDs?

The input pins should be 0010, while the output LEDs are 11111111 (with O7 as the MSB and O0 as the LSB) which is the binary equivalent of ff_{16}.



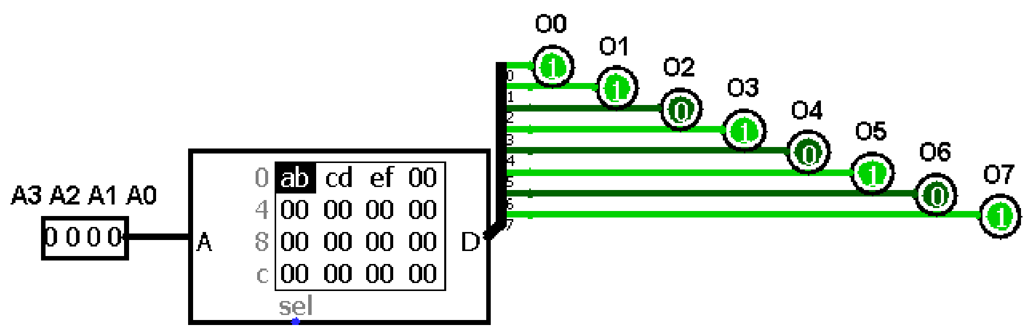
b. (1pt) For item 14 of Section 2.1, explain how you met the requirement – what cells or entries in the ROM needed to be programmed, and to what values. Include a picture of the circuit and attach the Logisim file (cs20lab10_1b.circ) of the circuit.

The word 0xCFECAF is made up of 24 bits. To store it in the ROM, only 3 cells were used since 1 cell is equivalent to a byte or 8 bits (3x8 =24). Since the word is in little endian, the LSB will be in the lowest numbered byte followed by the next bits. Hence, address 0000 contains 0xaf, address 0001 contains 0xec, and address 0010 contains 0xcf.



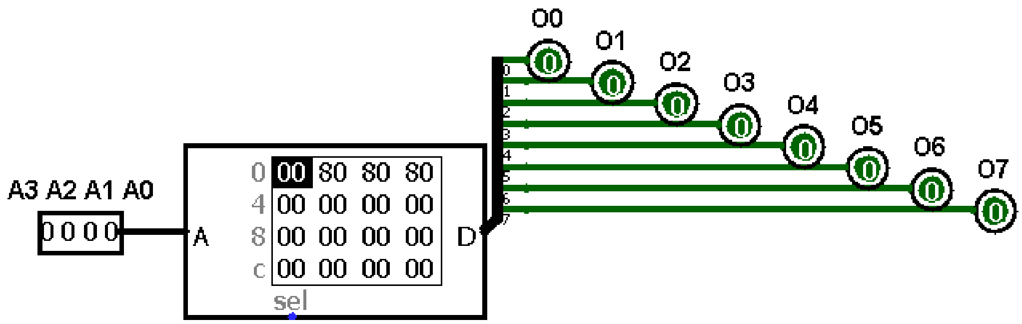
c. (1pt) For item 15 of Section 2.1, explain how you met the requirement – what cells or entries in the ROM needed to be programmed, and to what values. Include a picture of the circuit and attach the Logisim file (cs20lab10_1c.circ) of the circuit.

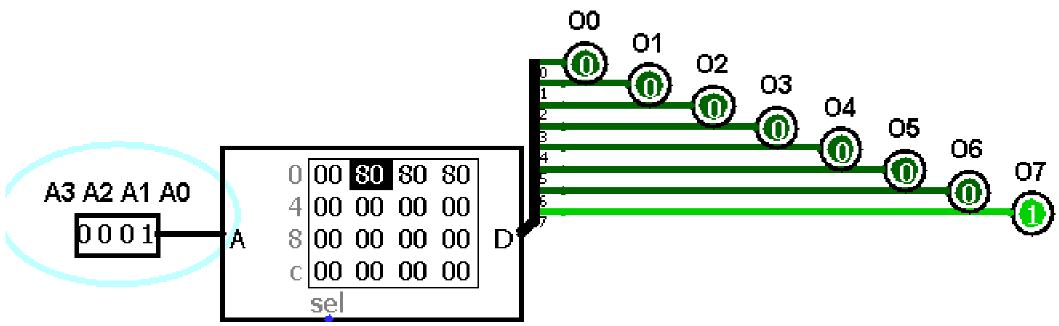
The word 0xABCDEF is made up of 24 bits. To store it in the ROM, only 3 cells were used since 1 cell is equivalent to a byte or 8 bits (3x8 =24). Since the word is in big endian, the MSB will be in the lowest numbered byte followed by the next bits. Hence, address 0000 contains 0xab, address 0001 contains 0xcd, and address 0010 contains 0xef.

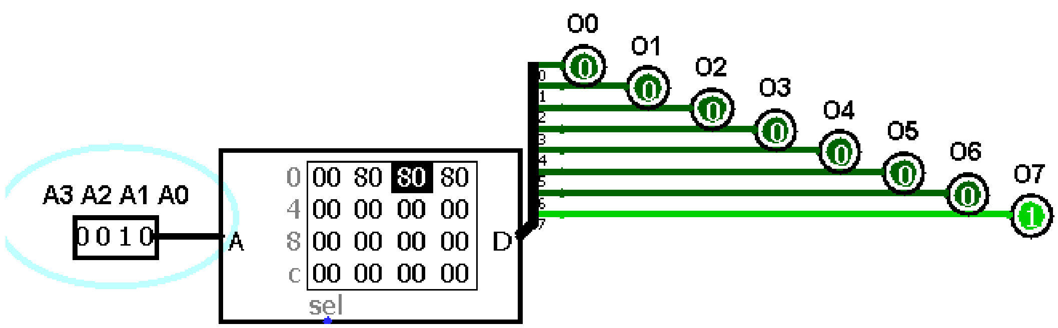


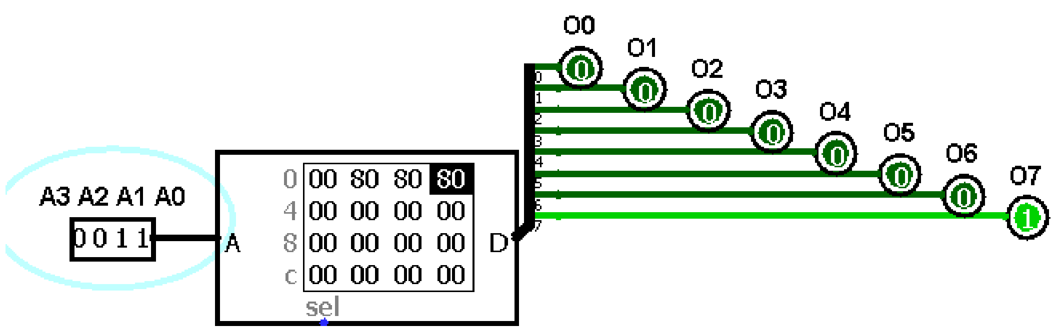
2. . (2pts) Logic gates using memory (look-up table)

a. (2pts) For Section 2.2, include pictures showing/proving that the ROM chip indeed acts as a 2-input OR gate. We expect 4 pictures in total - make sure that these pictures/ are well-labelled. Attach also the labelled Logisim file (cs20lab10_2.circ) of the circuit.


O0
O1
O2
O3
O4
O5
O6
O7
A3 A2 A1 A0
0 0 0 0
A
D
sel


O0
O1
O2
O3
O4
O5
O6
O7
A3 A2 A1 A0
0 0 0 1
A
D
sel


O0
O1
O2
O3
O4
O5
O6
O7
A3 A2 A1 A0
0 0 1 0
A
D
sel


O0
O1
O2
O3
O4
O5
O6
O7
A3 A2 A1 A0
0 0 1 1
A
D
sel

3. (4pts) Complex logic functions using memory (look-up table)

   a. (2pts) Explain how you met the requirement in Section 2.3 - what cells or entries in the ROM needed to be programmed, and to what values. For each ROM entry edited, explain its programmed content in relation to the BCD output representing the product of the two inputs, input1 and input2.

   In order to visualize the 2-bit multiplier, I created a table with corresponding decimal and hexadecimal equivalents.

Input 1		Input 2		Decimal	Output				Reversed	Hexadecimal
A3	A2	A1	A0	Output	O0	O1	O2	O3	Output	equivalent
0	0	0	0	0	0	0	0	0	0000	0
0	0	0	1	0	0	0	0	0	0000	0
0	0	1	0	0	0	0	0	0	0000	0
0	0	1	1	0	0	0	0	0	0000	0
0	1	0	0	0	0	0	0	0	0000	0
0	1	0	1	1	0	0	0	1	1000	8
0	1	1	0	2	0	0	1	0	0100	4
0	1	1	1	3	0	0	1	1	1100	C
1	0	0	0	0	0	0	0	0	0000	0
1	0	0	1	2	0	0	1	0	0100	4
1	0	1	0	4	0	1	0	0	0010	2
1	0	1	1	6	0	1	1	0	0110	6
1	1	0	0	0	0	0	0	0	0000	0
1	1	0	1	3	0	0	1	1	1100	C
1	1	1	0	6	0	1	1	0	0110	6
1	1	1	1	9	1	0	0	1	1001	9

For addresses 0000, 0001, 0010, 0100, 1000, and 1100 – since there is a 00 input, their product should be 0 hence, their values would also be 00.

As for address 0101 which represents 01_2*01_2, the answer should be 0001_2 (1_{10}). However, O0 acts as the MSB of the decoder, so the input of the decoder would be reverse the output (until O3) of the ROM. Hence, instead of 0001_2 , the value of the ROM will become 1000_2 which is 0x08.

The same thought process is done with the non-zero products. For address 0110 which represents 01_2*10_2 = 0010_2 (2_{10}), the reverse would be 0100_2 which is 0x04.

For address 0111 which represents 01_2*11_2 = 0011_2 (3_{10}), the reverse would be 1100_2 which is 0x0c.

For address 1001 which represents $10_2*01_2 = 0010_2$ (2_{10}), the reverse would be 0100_2 which is 0x04.

For address 1010 which represents $10_2*10_2 = 0100_2$ (4_{10}), the reverse would be 0010_2 which is 0x02.

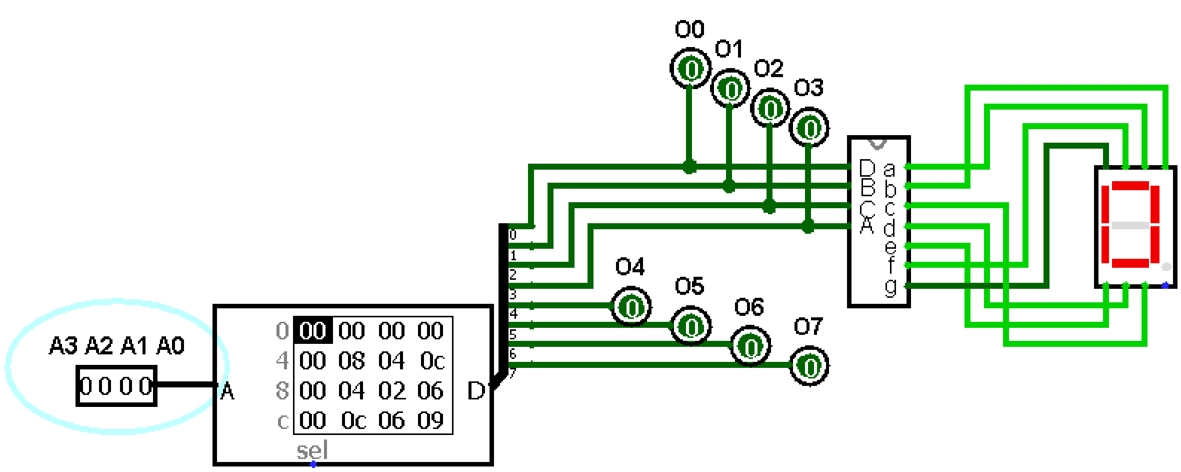
For address 1011 which represents $10_2*11_2 = 0110_2$ (6_{10}), the reverse would still be 0110_2 which is 0x06.

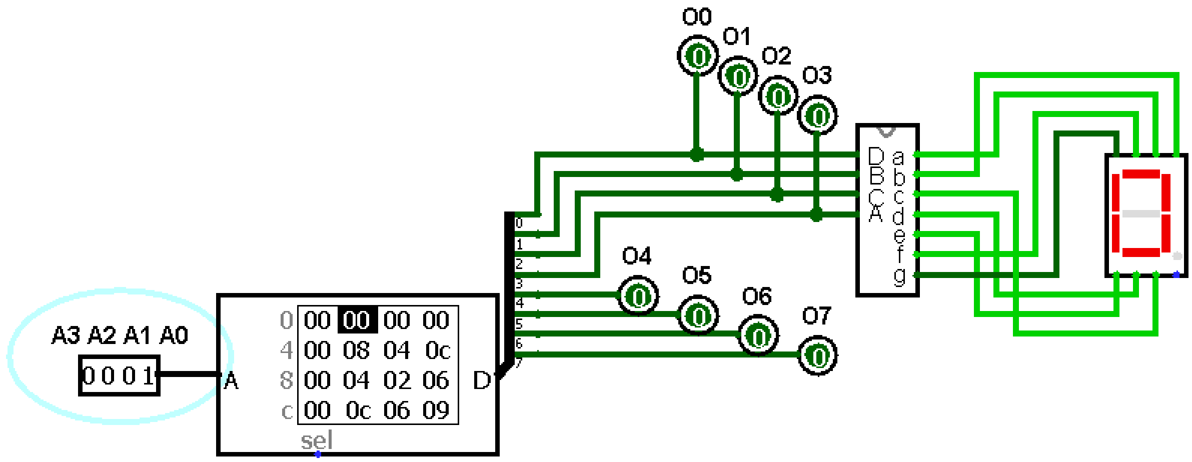
For address 1101 which represents $11_2*01_2 = 0011_2$ (3_{10}), the reverse would be 1100_2 which is 0x0c.

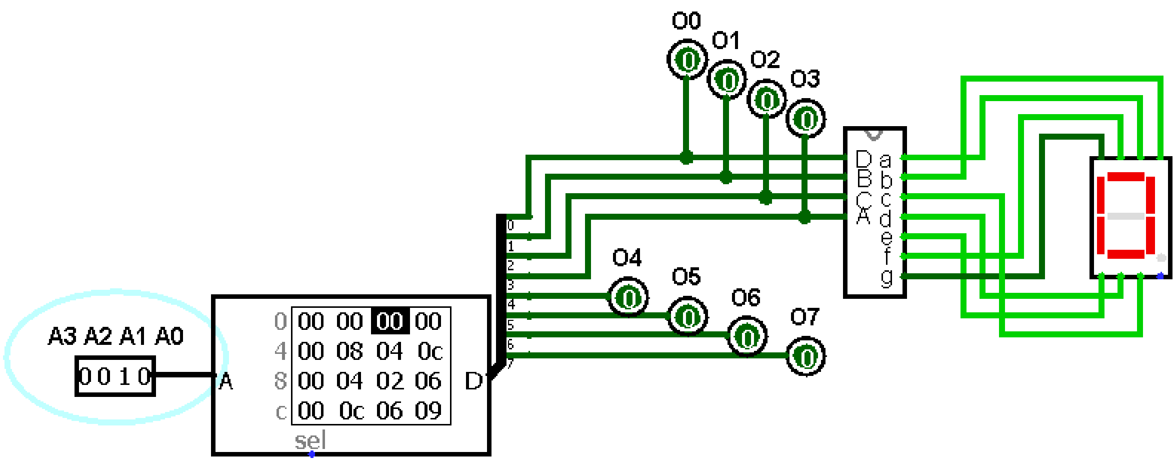
For address 1110 which represents $11_2*10_2 = 0110_2$ (6_{10}), the reverse would still be 0110_2 which is 0x06.

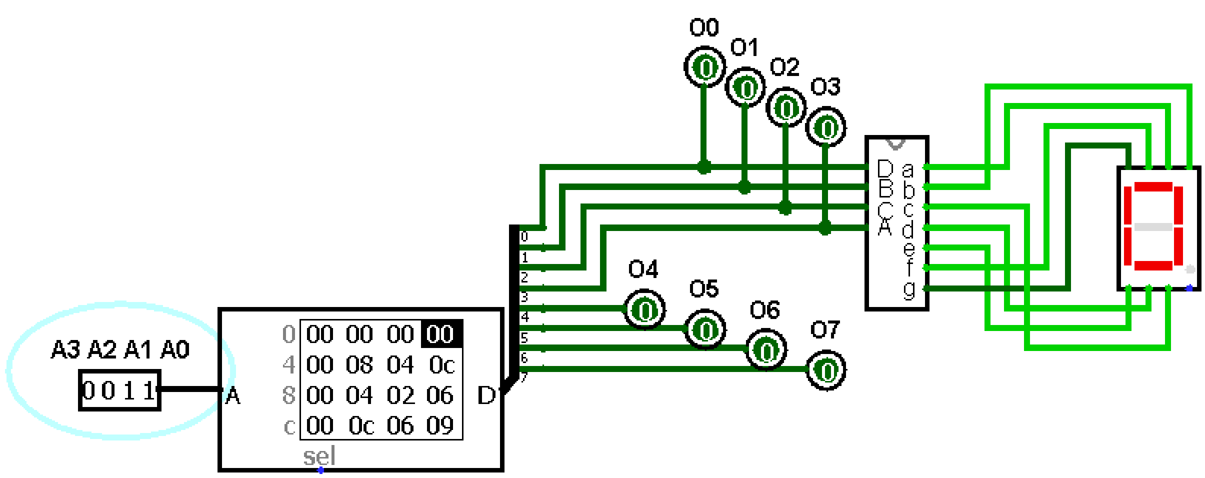
Finally, for address 1111 which represents $11_2*11_2 = 1001_2$ (9_{10}), the reverse would still be 1001_2 which is 0x09.

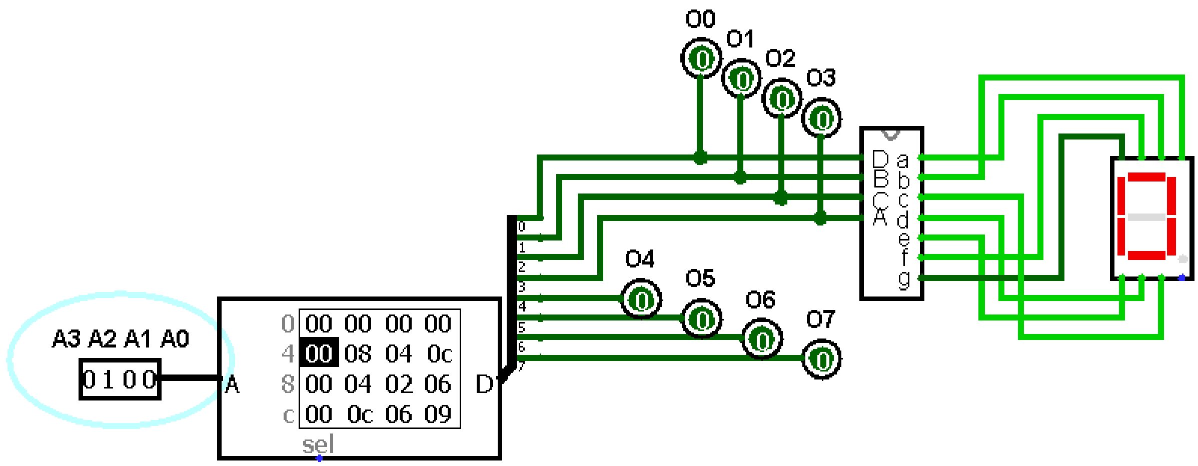
b. (2pts) For Section 2.3, include pictures showing/proving that the ROM chip indeed acts as a 2-bit multiplier. We expect 16 pictures in total - make sure that these pictures are well-labelled (what was the input at that state, etc.). Attach also the labelled Logisim file (cs20lab10_3.circ) of the circuit.

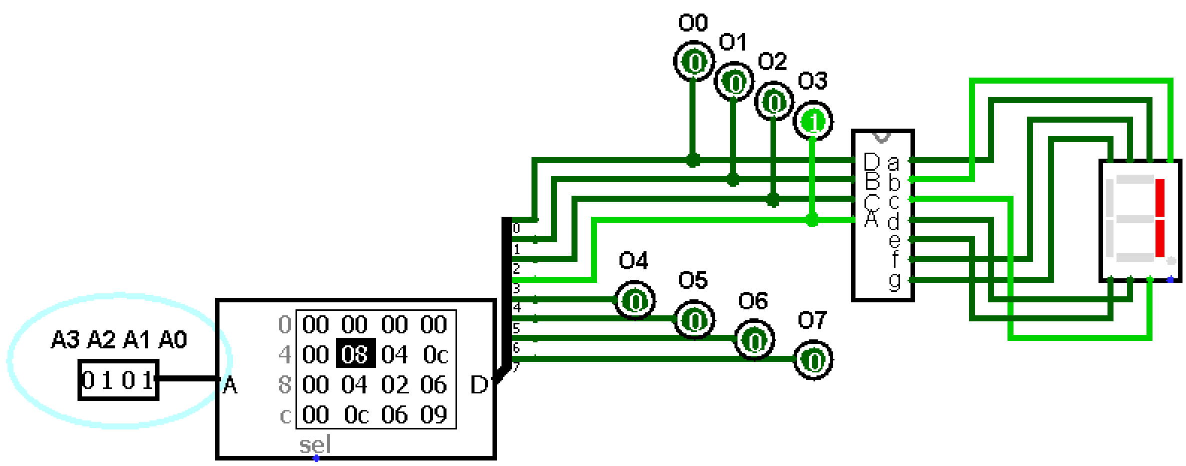


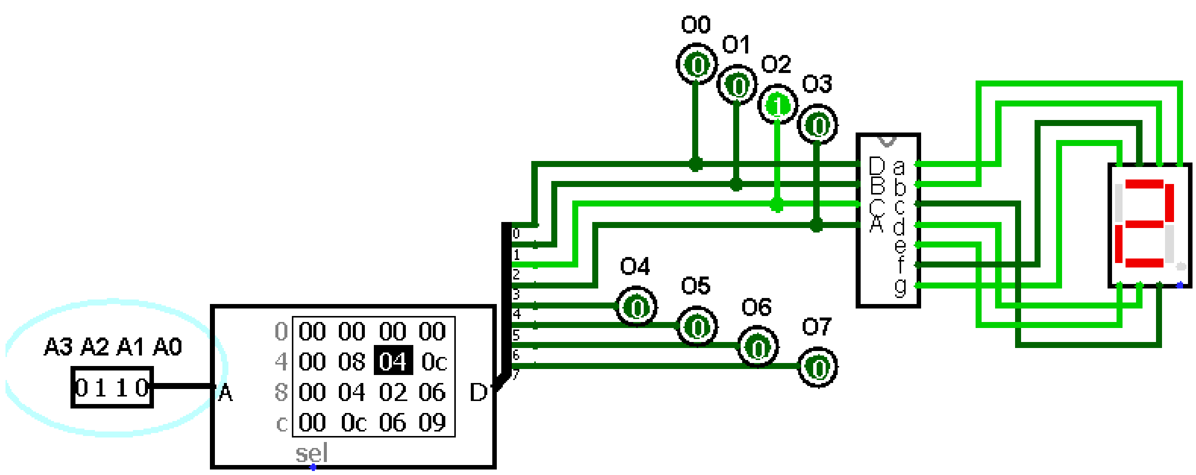

O0
O1
O2
O3
O4
O5
O6
O7
A3 A2 A1 A0
0 0 0 1
0 00 00 00 00
4 00 08 04 0c
8 00 04 02 06
c 00 0c 06 09
sel
A
D

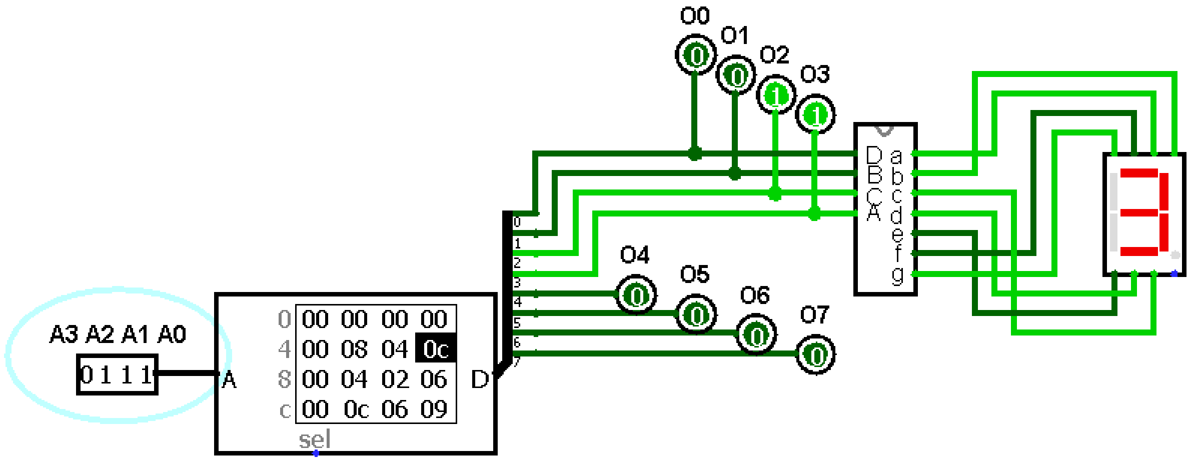

O0
O1
O2
O3
O4
O5
O6
O7
A3 A2 A1 A0
0 0 1 0
0 00 00 00 00
4 00 08 04 0c
8 00 04 02 06
c 00 0c 06 09
sel
A
D

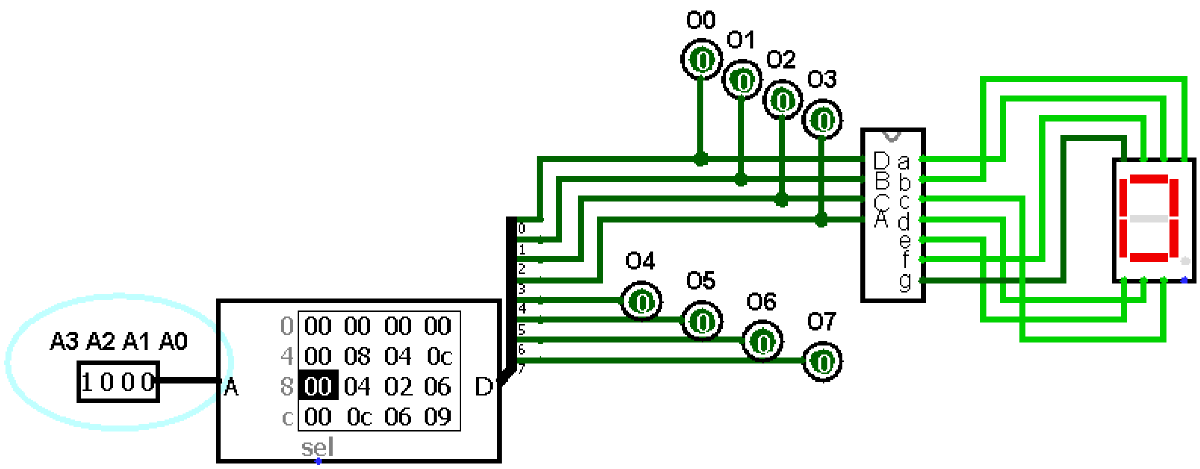

O0
O1
O2
O3
O4
O5
O6
O7
A3 A2 A1 A0
0 0 1 1
0 00 00 00 00
4 00 08 04 0c
8 00 04 02 06
c 00 0c 06 09
sel
A
D

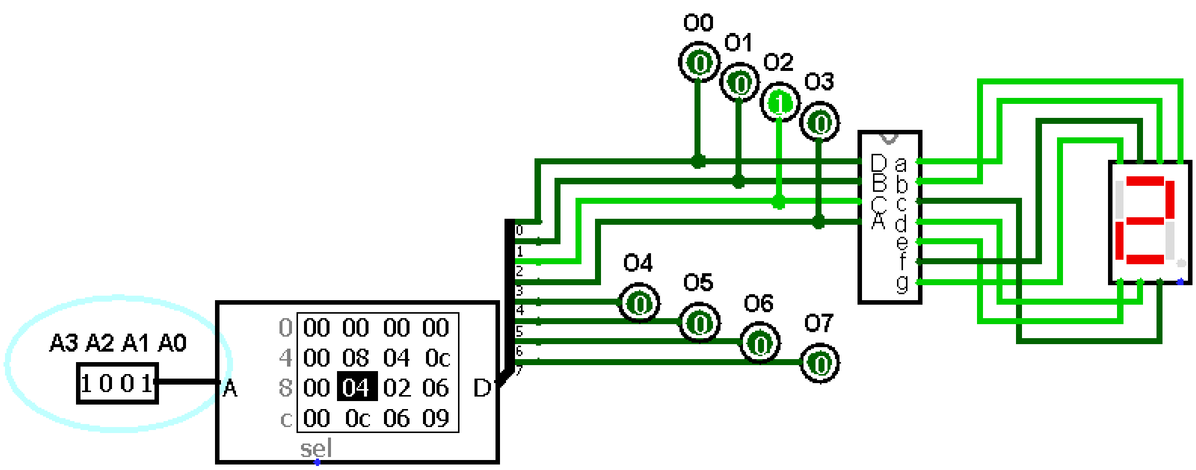

A3 A2 A1 A0
0100
00 00 00 00
00 08 04 0c
00 04 02 06
00 0c 06 09
sel

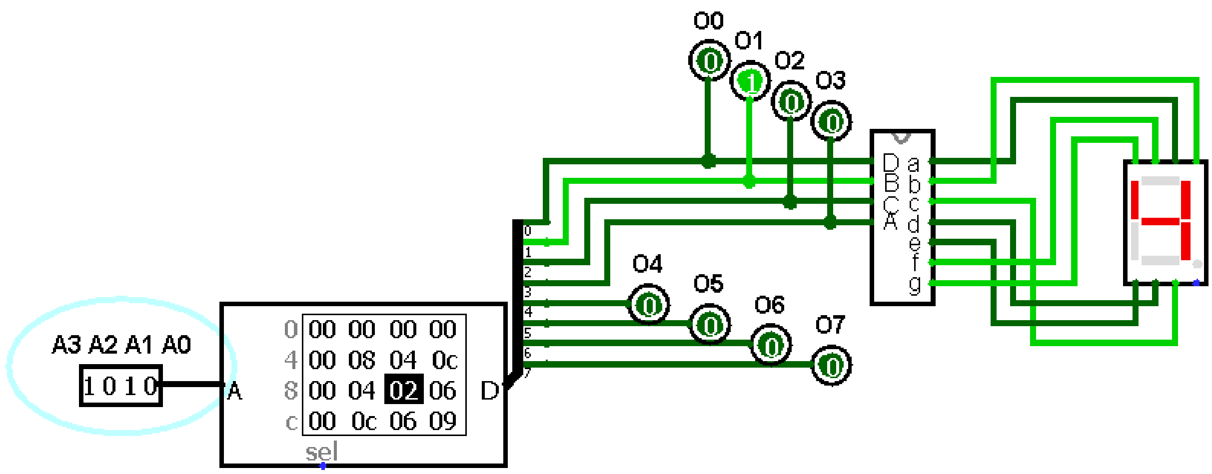

A3 A2 A1 A0
0101
00 00 00 00
00 08 04 0c
00 04 02 06
00 0c 06 09
sel

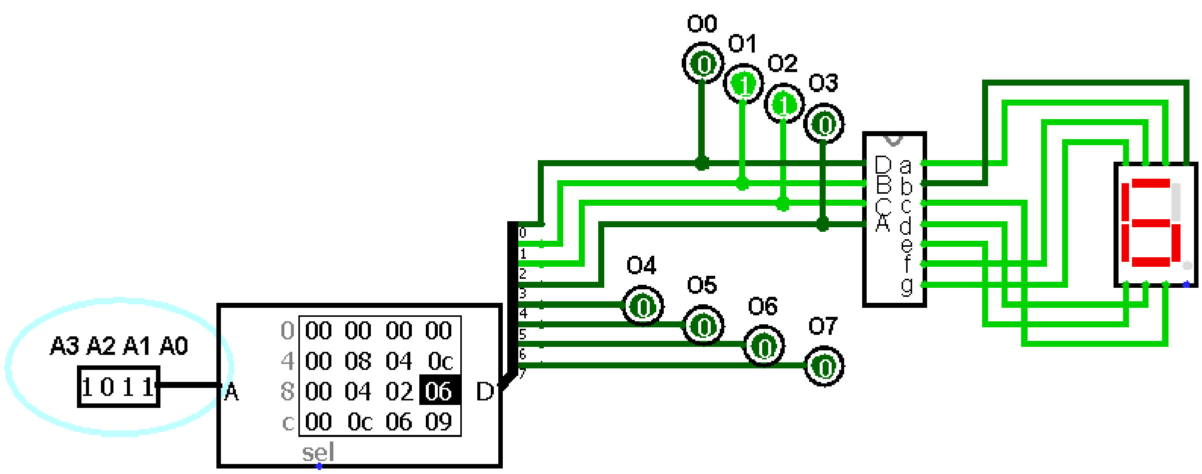

A3 A2 A1 A0
0110
00 00 00 00
00 08 04 0c
00 04 02 06
00 0c 06 09
sel

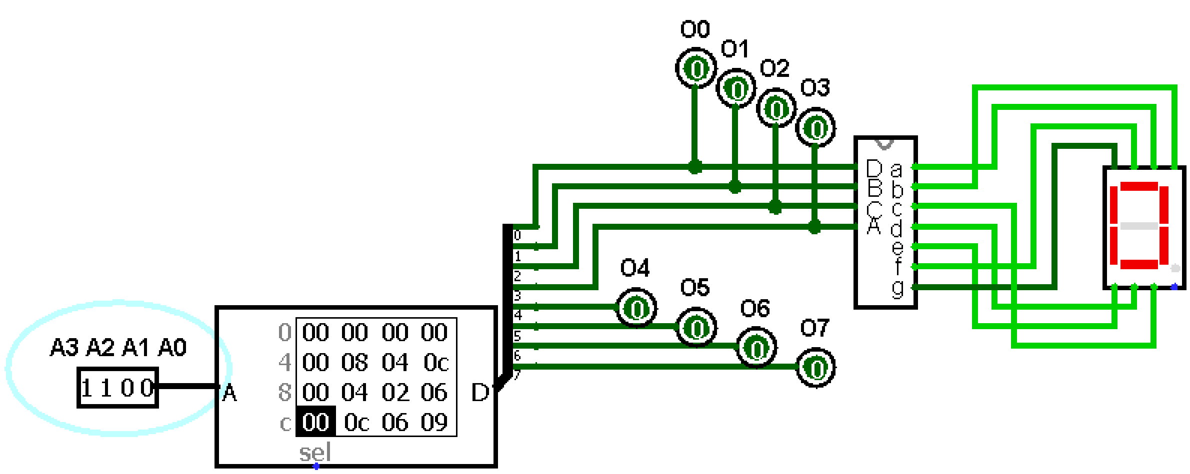

O0
O1
O2
O3
O4
O5
O6
O7
A3 A2 A1 A0
0 1 1 1
A
D
sel
0 00 00 00 00
4 00 08 04 0c
8 00 04 02 06
c 00 0c 06 09
D B C A
a b c d e f g

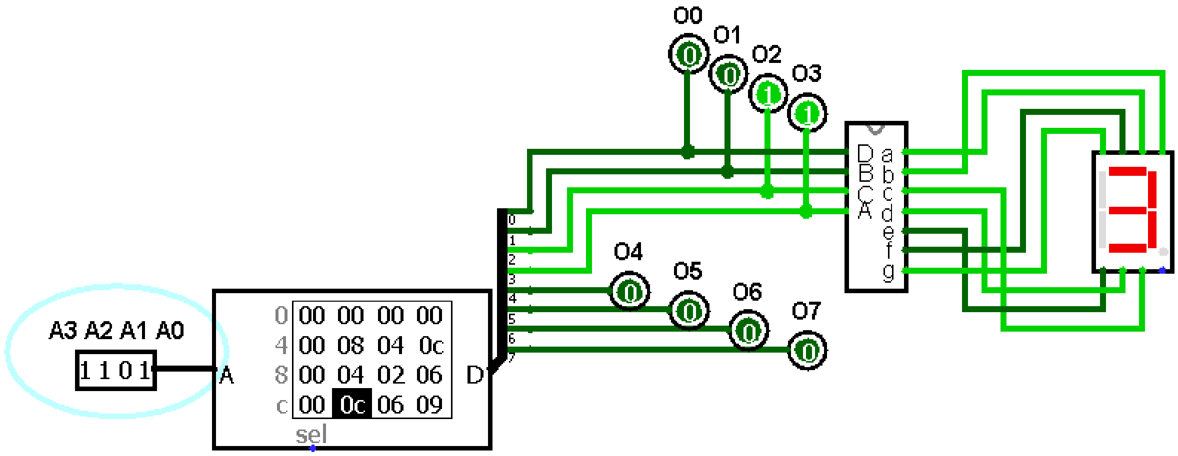

O0
O1
O2
O3
O4
O5
O6
O7
A3 A2 A1 A0
1 0 0 0
A
D
sel
0 00 00 00 00
4 00 08 04 0c
8 00 04 02 06
c 00 0c 06 09
D B C A
a b c d e f g

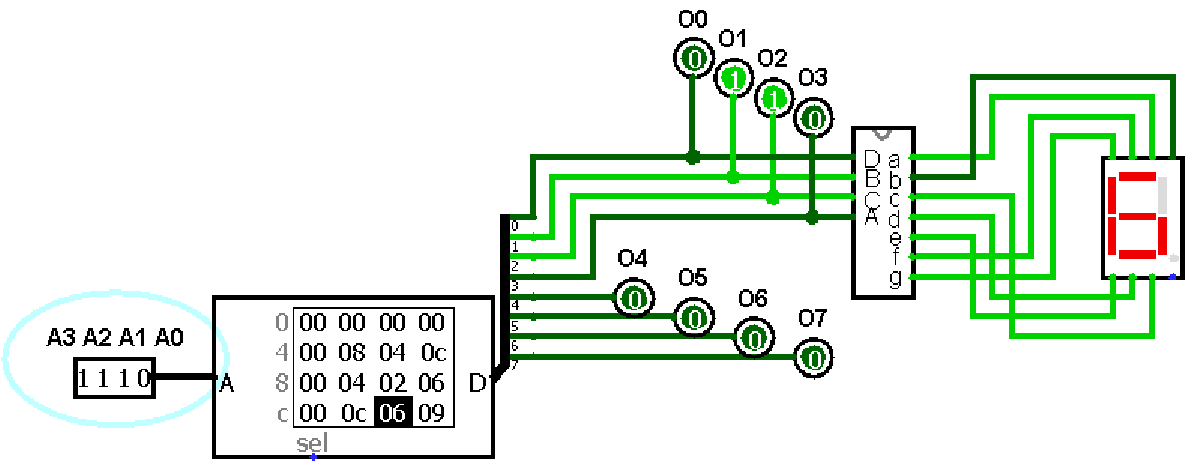

O0
O1
O2
O3
O4
O5
O6
O7
A3 A2 A1 A0
1 0 0 1
A
D
sel
0 00 00 00 00
4 00 08 04 0c
8 00 04 02 06
c 00 0c 06 09
D B C A
a b c d e f g

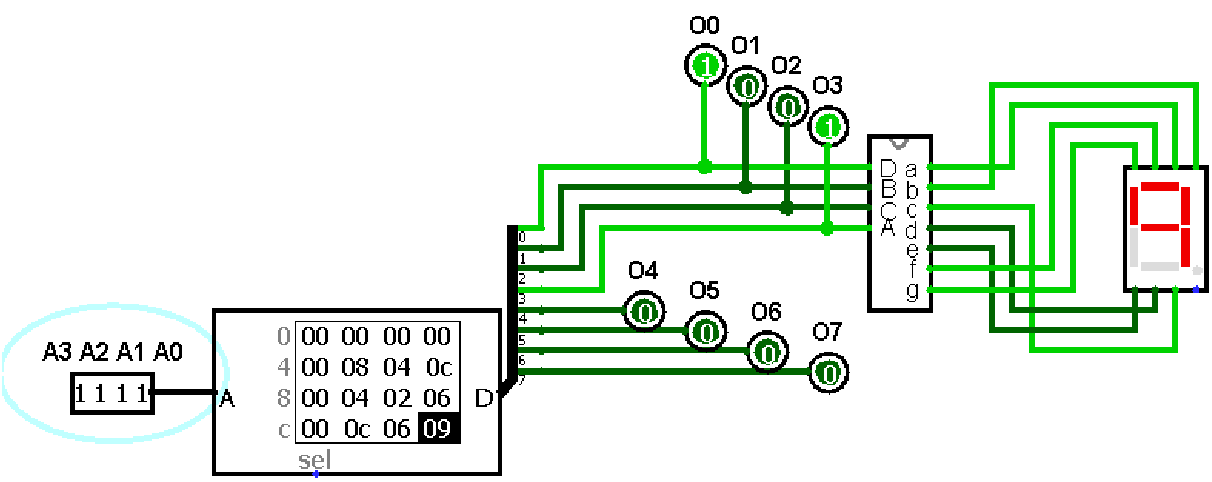

O0
O1
O2
O3
D B C A
a b c d e f g
O4
O5
O6
O7
A3 A2 A1 A0
1 0 1 0
A
0 00 00 00 00
4 00 08 04 0c
8 00 04 02 06
c 00 0c 06 09
D
sel


O0
O1
O2
O3
D B C A
a b c d e f g
O4
O5
O6
O7
A3 A2 A1 A0
1 0 1 1
A
0 00 00 00 00
4 00 08 04 0c
8 00 04 02 06
c 00 0c 06 09
D
sel


O0
O1
O2
O3
D B C A
a b c d e f g
O4
O5
O6
O7
A3 A2 A1 A0
1 1 0 0
A
0 00 00 00 00
4 00 08 04 0c
8 00 04 02 06
c 00 0c 06 09
D
sel


O0
O1
O2
O3
O4
O5
O6
O7
A3 A2 A1 A0
1 1 0 1
0 00 00 00 00
4 00 08 04 0c
8 00 04 02 06
c 00 0c 06 09
sel
A
D


O0
O1
O2
O3
O4
O5
O6
O7
A3 A2 A1 A0
1 1 1 0
0 00 00 00 00
4 00 08 04 0c
8 00 04 02 06
c 00 0c 06 09
sel
A
D


O0
O1
O2
O3
O4
O5
O6
O7
A3 A2 A1 A0
1 1 1 1
0 00 00 00 00
4 00 08 04 0c
8 00 04 02 06
c 00 0c 06 09
sel
A
D