Homework 3 - CSS422

Vu Dinh - 1440064

Question 1.

1. Each device has 11 address lines, therefore 2^{11} addressable memory locations, each of which can hold 16 bits of data. The capacity of each device is thus: $2^{11} \times 2^{4} = 2^{15}$ (bits)

2. Since 32-bit data bus is required, these devices have to be arranged into 2 chips horizontally; this necessitates 1 chip selection bit. There are 11 offset bits to interface the system with each device, leaving $18 - 1 - 11 = 6$ address lines for page-selection.

6	11	1

   Since the memory system has word-addressable capability, it does not fully support the capacity stated in its specifications, i.e. $2^{18} \times 2^{5} = 2^{23}$. Instead, it can only support 2^{18} unique addressable spaces if each space only holds 16-bits of data; when 32-bits of data need to be stored, the system made with these devices can only support 2^{17} unique spaces. See this discussion for details.

3. Each page has 2^{11} unique addresses in each device, so a total of 2^{12} unique addresses per page. Page 0 thus occupies addresses from 000000000000000000 to 000000111111111111 or from $00000 to $00FFF

Page (decimal)	Starting Address (hex)	Ending Address (hex)
1	$01000	$01FFF
50	$32000	$32FFF
100	unavailable	unavailable

   HOWEVER, the instruction clearly says to "assume that each page of memory is the same size as the address range of one memory device", which doesn't make sense. Under that assumption, though, there are 2^{11} unique addresses per page, so page 0 occupies from 000000000000000000 to 000000011111111111 or from $00000 to $007FF

Page (decimal)	Starting Address (hex)	Ending Address (hex)
1	$00800	$00FFF
50	$19000	$197FF
100	$32000	$327FF

4. Each device used in the system can only support word-level memory addressing since its data bus is 16-bit wide. If:
   - the devices being used in the system had an 8-bit data bus, or
   - the 16-bit data devices themselves were composed of 8-bit data bus devices

   then it may be possible for the memory system to have byte-addressable memory locations.
   Neither of these conditions was specified; it is unwise to assume that they are true. The system is, thus, assumed to **not** have addressing memory locations at the byte-level.

Question 2.

1. To compose a 32-bit data bus with devices using 8-bit data bus, we need 4 devices arranged horizontally per page, so we need 2 chip selection bits. The address partitioning is thus as follows:

10	17	2

2. According to the partitioning table, we need $2^{10} = 1024$ pages of devices, each with 4 devices, so a total of $2^{12} = 4096$ devices.
3. Each page has 4 devices, each with 2^{17} address spaces, so a total of $2^{17} \times 2^{2} = 2^{19}$ unique address spaces per page. That is, page 0 occupies from 00000000000000000000000000000 to 00000000001111111111111111111, or in hexadecimal: from $00000000 to $0007FFFF

Page (decimal)	Starting Address (hex)	Ending Address (hex)
15	$00780000	$007FFFFF
387	$0C180000	$0C1FFFFF
1020	$1FE00000	$1FE7FFFF

Question 3.

Logical equations for read/write operations:

- long-word read = CE * OE * ~WE0 * ~WE1 * ~WE2 * ~WE3

~CE	~OE	~WE0	~WE1	~WE2	~WE3
active	active	inactive	inactive	inactive	inactive

- byte-write to address 01 in big endian: CE * ~OE * ~WE0 * WE1 * ~WE2 * ~WE3

~CE	~OE	~WE0	~WE1	~WE2	~WE3
active	inactive	inactive	active	inactive	inactive

- word-write to address 00 in little endian: CE * ~OE * ~WE0 * ~WE1 * WE2 * WE3

~CE	~OE	~WE0	~WE1	~WE2	~WE3
active	inactive	inactive	inactive	active	active

- word-read from address 10: CE * OE * ~WE0 * ~WE1 * ~WE2 * ~WE3

~CE	~OE	~WE0	~WE1	~WE2	~WE3
active	active	inactive	inactive	inactive	inactive

Question 4.

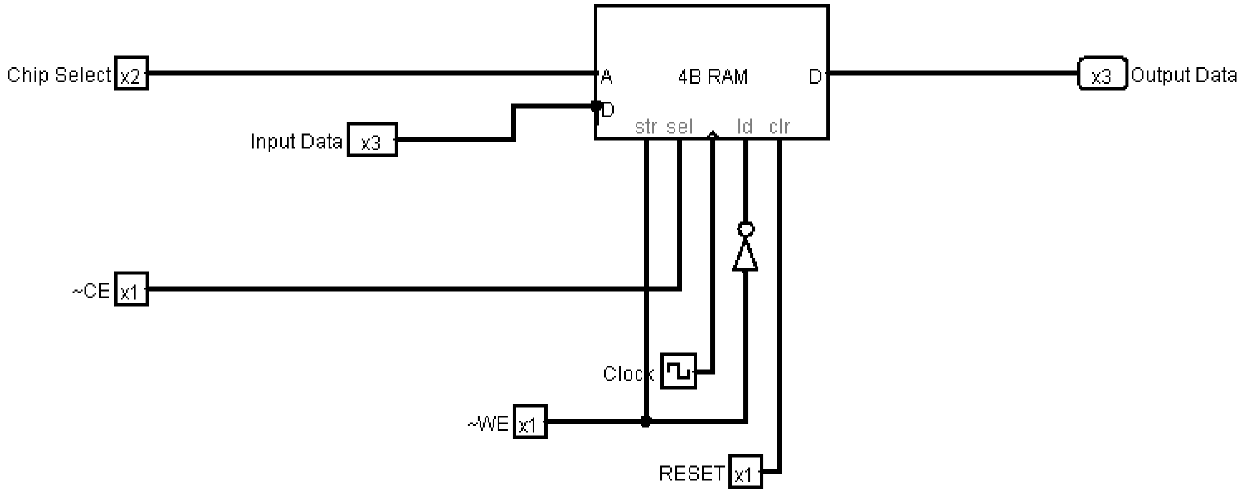
1. Write data [101] to address 3

RESET	S1	S0	Bit2	Bit1	Bit0	~WE
LOW	HIGH	HIGH	HIGH	LOW	HIGH	HIGH

2. Read data from address 1

RESET	S1	S0	Bit2	Bit1	Bit0	~WE
LOW	LOW	HIGH	X	X	X	LOW

3. Circuit diagram



Problem 5.

The sequence of events of interest is:

- #data6 (%01001111 or $4F) is moved to D1
- #data7 (%00010111 or $17) is moved to D2
- #data3 ($5555) is moved to D3
- ---
- #addr1 ($4000) is moved to A0
- D1 ($4F) is moved to A0’s indirect address, i.e. $4000.
    - A0’s content is incremented by 1, i.e. A0 now holds $4001
- D2 is moved to A0’s indirect address, i.e. $4001.
- ---
- #addr1 ($4000) is moved to A1
- A logical AND operation is performed on the data stored in D3 and A1’s indirect address, i.e. $4000
    - D3 stores $5555, or %101010101010101
    - $4000 stores $4F, or %01001111, which is exactly one byte of data
    - $4001 stores $17, or %00010111, which is exactly one byte of data
    - Together, the WORD value at $4000 is %0100111100010111
    - AND the following pair of values:
        - %0101010101010101
        - %0100111100010111
        - -------AND-------
        - %0100010100010101 , which is $4515

The content at $4000 after the AND operation is thus $4515.