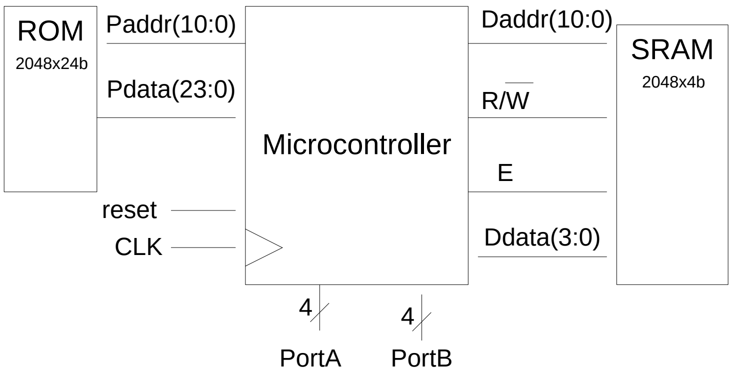
I. Background

The block diagram of the microcontroller is shown below, along with the original assembly language instruction set and modifications, detailed diagram, and function tables:



<u>ALU Commands</u>:

Function	**Source A**	**Source B**	**Destination**
ADD	reg (R0-R3)	reg (R0-R2) or NI[*]	reg (R0-R3) or [mem addr] or Port A
SUB			
XOR			
AND			
OR			

[*] NI (Numerical Input) cannot be used with [mem addr] destination

Function	**Source A**	**Destination**
INV	reg (R0-R3)	reg (R0-R3) or [mem addr] or PortA

<u>Data Commands</u>:

Command	**Source**	**Destination**
LD	[mem addr]	reg (R0-R3) or Port A
STR	reg (R0-R3) or Port B	[mem addr]
MOV	reg (R0-R3) or Port B[†] or NI	reg (R0-R3) or Port A[†]

[†] Port A and Port B are mutually exclusive

Hardware Structure:

Program memory: 2048 × 24 bits

Instruction pre-fetching for branch instructions

- PC ← PC + 1 each clock cycle except for branch instructions
- Branch instructions should be prefetched, such that they only require one clock cycle
- Program addresses 1 – 15 are reserved for interrupts, so they cannot be used by you

Data memory: not needed for this test

Port A: 4-bit output port

Port B: 4-bit input port

Flag register: 4 bits:

OV (overflow): updated each addition operation (i.e., set if $A + B > 4$ bits) or subtraction operation (i.e., set if $A < B$)

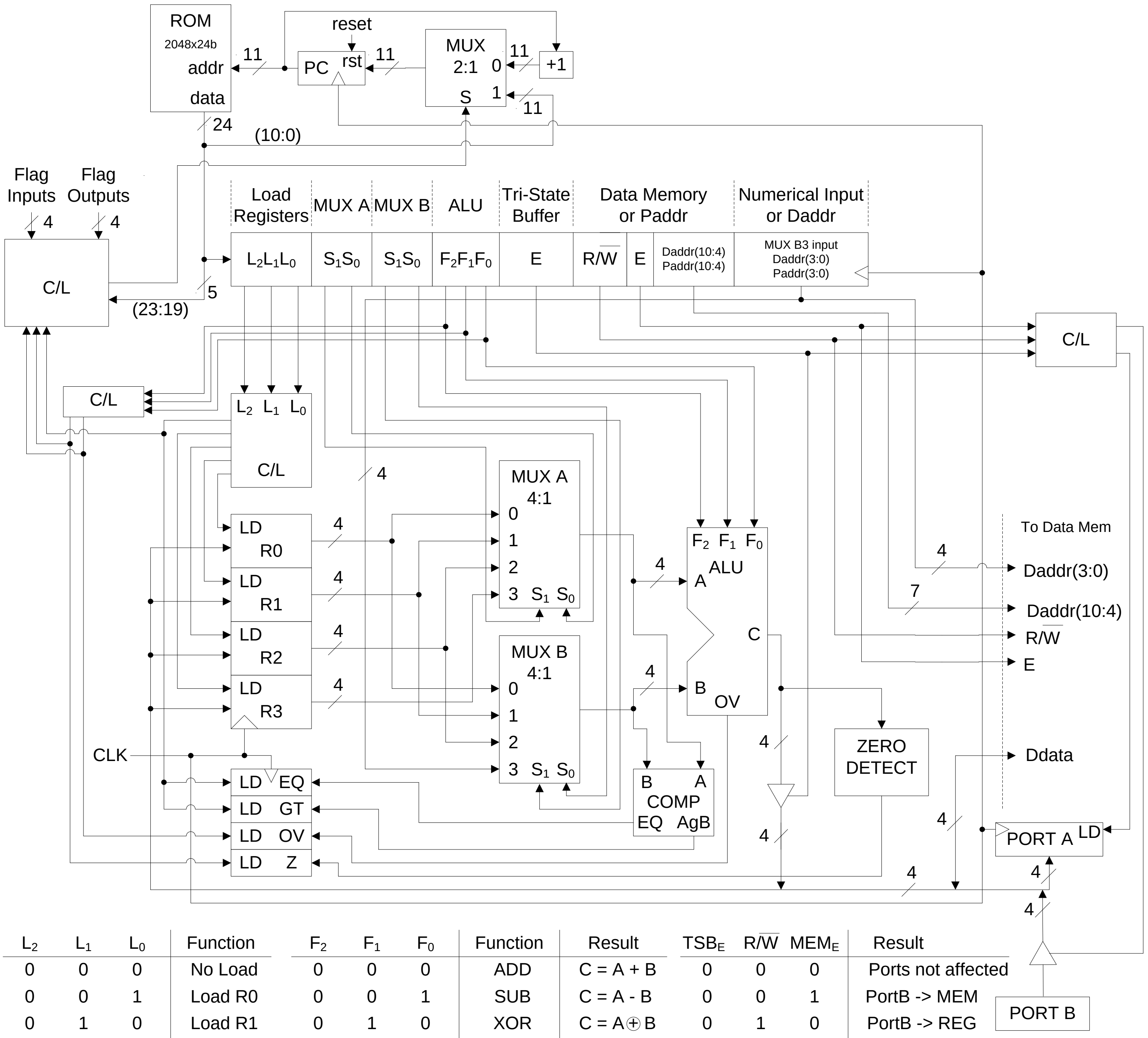
EQ (equal), GT (greater than): updated each compare operation

Z (C = zero): updated each ALU operation, not including Pass A or Pass B (i.e., 000-101)

Additional Instructions:

NOP (no operation is performed and the state of the machine doesn’t change)
CMP source A source B (compare A with B, set EQ and GT bits): $L_2L_1L_0 = 101$
BOV [address] (PC ← Paddr if OV bit set): $L_2L_1L_0 = 110$, MUX_A $S_1S_0 = 00$
BEQ [address] (PC ← Paddr if EQ bit set): $L_2L_1L_0 = 110$, MUX_A $S_1S_0 = 01$
BGT [address] (PC ← Paddr if GT bit set): $L_2L_1L_0 = 110$, MUX_A $S_1S_0 = 10$
BZ [address] (PC ← Paddr if Z bit set): $L_2L_1L_0 = 110$, MUX_A $S_1S_0 = 11$
BNE [address] (PC ← Paddr if EQ bit not set): $L_2L_1L_0 = 111$, MUX_A $S_1S_0 = 00$
BLT [address] (PC ← Paddr if GT bit not set and EQ bit not set): $L_2L_1L_0 = 111$, MUX_A $S_1S_0 = 01$
BNZ [address] (PC ← Paddr if Z bit not set): $L_2L_1L_0 = 111$, MUX_A $S_1S_0 = 10$
BR [address] (PC ← Paddr): $L_2L_1L_0 = 111$, MUX_A $S_1S_0 = 11$

- $F_2F_1F_0 = 11x$ for LD, STR, MOV, NOP, CMP, and Bxx instructions



L_2	L_1	L_0	Function	F_2	F_1	F_0	Function	Result	TSB_E	R/$\overline{W}$	MEM_E	Result
0	0	0	No Load	0	0	0	ADD	C = A + B	0	0	0	Ports not affected
0	0	1	Load R0	0	0	1	SUB	C = A - B	0	0	1	PortB -> MEM
0	1	0	Load R1	0	1	0	XOR	C = A⊕B	0	1	0	PortB -> REG
0	1	1	Load R2	0	1	1	OR	C = A∨B	0	1	1	MEM -> PortA iff $L_2L_1L_0$ = 000
1	0	0	Load R3	1	0	0	AND	C = A∧B	1	0	0	Ports not affected
1	0	1	Load EQ, GT	1	0	1	INV	C = $\overline{A}$	1	0	1	Ports not affected
1	1	0	No Load	1	1	0	Pass B	C = B	1	1	0	ALU -> PortA
1	1	1	No Load	1	1	1	Pass A	C = A	1	1	1	Ports not affected

II. Examples:

The following examples will help you understand the structure of the micro-controller, and the basic syntax of assembly programming and C programming.

1. The basic syntax of assembly language is.   “( )” means it’s optional
(label:)   mnemonic   (operands)   (;comments)

2. A sample instruction with its binary opcode
ADD   R3   R0   [11001001010]   ; memory address [11001001010]←R3+R0
Binary opcode: 000 11 00 000 101 11001001010
Meaning of opcode: No load; R3; R0; ADD; ports not affected; memory address

3. The following assembly program demonstrates how to use a loop to generate a delay of 32 ms. It’s assumed that the frequency of the clock is 1 KHz.

```
            CALL DELAY   ; call the delay subroutine
            END          ; end of main program
;----------delay subroutine------------------
DELAY:      MOV #7 R0    ; the loop will be repeated for 7 times. #sign indicating numerical input
LOOP:       NOP          ; 1 cycle
            NOP          ; 1 cycle
            SUB R0 #1 R0 ; 1 cycle, R0←R0-1
            BNZ LOOP     ; 1 cycle. Result is not zero, jump back to the start of the loop
            NOP          ; 1 cycle
            RET          ; the end of the subroutine, return to the instruction following
                         ; call instruction.
;---------end of subroutine---------------------
```

Each loop contains 4 clk cycles. With the loop repeated for 7 times, the total number of clk cycles in the loop is 28. There’s also one clk cycle to call the subroutine, one clk cycle at the beginning of the subroutine to initialize R0, one clk cycle to return from the subroutine, and one additional NOP added after the loop, such that the entire subroutine call requires 32 clk cycles. Since the clock frequency is 1KHz, then the delay incurred by the loop is 32 ms. (Please note the micro-processor hardware described in the handout doesn’t support subroutines. However, for convenience, we will assume that subroutines are supported as in the example, and you will need to use subroutines to solve some of the problems).

4. The following C program demonstrates how to access the ports of the micro-controller. It’s assumed that the ports, PA and PB, are already defined in port.h. The program reads a single bit from port B bit 0, and sends a constant value 8 to port A.

```
#include <port.h>
void main(void)
{
    bit a;          // variable a is a single bit
    uint4 b = 8;    // variable b is a 4-bit unsigned integer
    While(1)
    {
        a = PB.0;
        PA = b;
    }
}
```

III. Test Questions

1. (10 points) Assume the data bus of a micro-controller is 24-bit. Therefore, each address points to a 24-bit data block in the RAM

(a) How many bytes are in a section of RAM with address range FF0D2A ~ FF117B? (5 points)

(b) A block of 3KB are stored in consecutive memory blocks inside a RAM. If the address of the first block is 0F0A, what is the address of the last block? (5 points)

2. Briefly describe the difference between an assembler and a compiler. (5 points)

3. (14 points) Encode the following assembly language instructions using the maximum number of don’t cares:

(a) ADD R3 R1 [10000100011] (7points)

(b) LD [01001100100] R3 (7 points)

3. (14 points) Decode the following assembly language instructions

(a) 1A6F27 (7 points)

(b) 1D8D9B (7 points)

4. (20 points) Port A of the microcontroller is connected to 4 LEDs, and Bit 0 of Port B is connected to a switch. Assume the clock frequency is 1MHz. Write an **assembly language program** to achieve the following functions:
(1) if the switch is on (Bit 0 of Port B is high), alternate the 4 LEDs based on the following patterns: 1010 (40 us) → 0101 (40 us) → 1010 (40 us) → 0101 (40 us) → …
(2) if the switch is off, turn off all the LEDs.
Don’t worry about the few extra clock cycles needed to check Port B; you only need to use one 40 us delay subroutine.

5. (20 points) Write an **assembly language** program to find the sum of 2 arbitrary 8-bit numbers, N1 + N2. N1 is read from port B in two consecutive clk cycles (least significant nibble first) and stored in R0 and R1. N2 is read from Port B in the two subsequent clk cycles (least significant nibble first) and stored in R2 and R3. Store the result at RAM locations 400H (least significant nibble), 401H (most significant nibble), and 402H (carry out). For example, N1 = 11001010_2 and N2 = 10110110_2 would yield the following result: [400H] = 0000_2, [401H] = 1000_2, and [402H] = 0001_2.

6. (17 points) Write a **C program** to continuously send the numbers 0 to 15 to port A. Bit 0 of port B is connected to an external debouncing switch. Every time the position of the switch is changed (on → off, or off → on), send one number to port A (in ascending order). After transmitting the last number, go back to the first number and repeat the above procedure.