TRACS

TRAnsition semester Computer System

Version 2.4 (2021)

Computer Specifications

- 8-bit data bus (8-bit memory)
- 16-bit instruction (two-cycle instruction fetch, big endian)
- Load & Store: memory to memory
- Instruction Set Architecture: CISC
- 24 single word instruction
- non-pipelined (instruction and data share bus)
- 8-bit ALU (supports signed arithmetic operations)

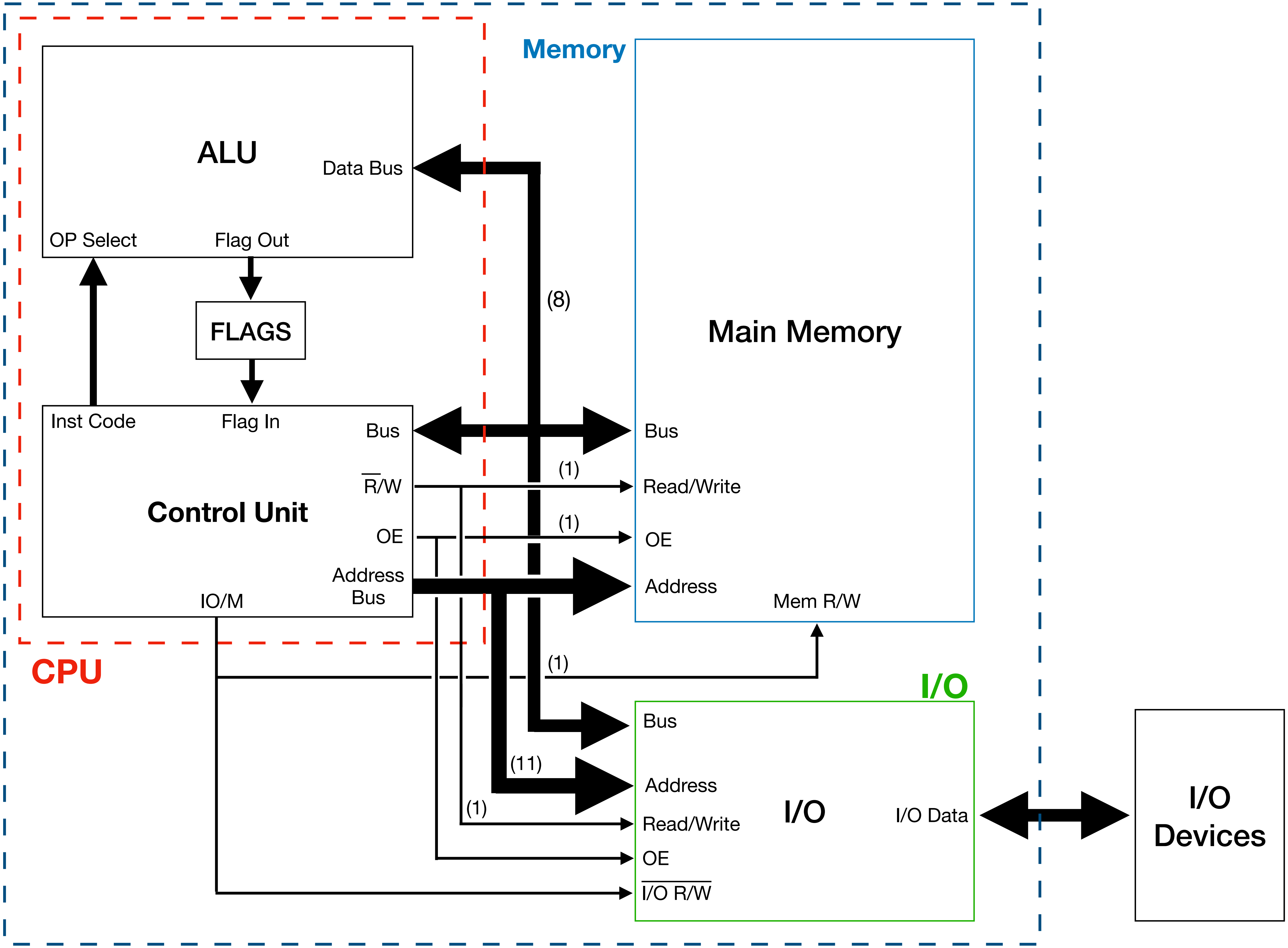

Figure 1. Computer Architecture

Signal Descriptions

- **BUS** - an 8-bit bus shared by both instruction and data
- **ADDR** (address bus) - an 11-bit bus for memory and I/O address
- **Control Signals**
  - **Inst Code** - channel for the 5-bit instruction code to the ALU
  - **IOM** - Main Memory or I/O Memory access control (‘1’-Memory, ‘0’-I/O)
  - **RW** - Memory read/write control (‘1’-write, ‘0’-read)
  - **OE** - enables data read/write on the Main Memory or I/O Memory (‘1’-enable, ‘0’-disable)

ALU

- 8-bit operands and 8-bit accumulator (ACC)
- The ACC returns to the ALU circuitry as the first operand with the BUS as the second operand
- Implicit operation (instruction has no operand) for *addition*, *subtraction*, *multiplication*, AND, OR, NOT, XOR, shift left & shift right where
  - *ACC <- ACC <operation> BUS*
- ALU circuitry controls the ACC (accumulator) module
- Does not support floating point
- ACC module sets the flags

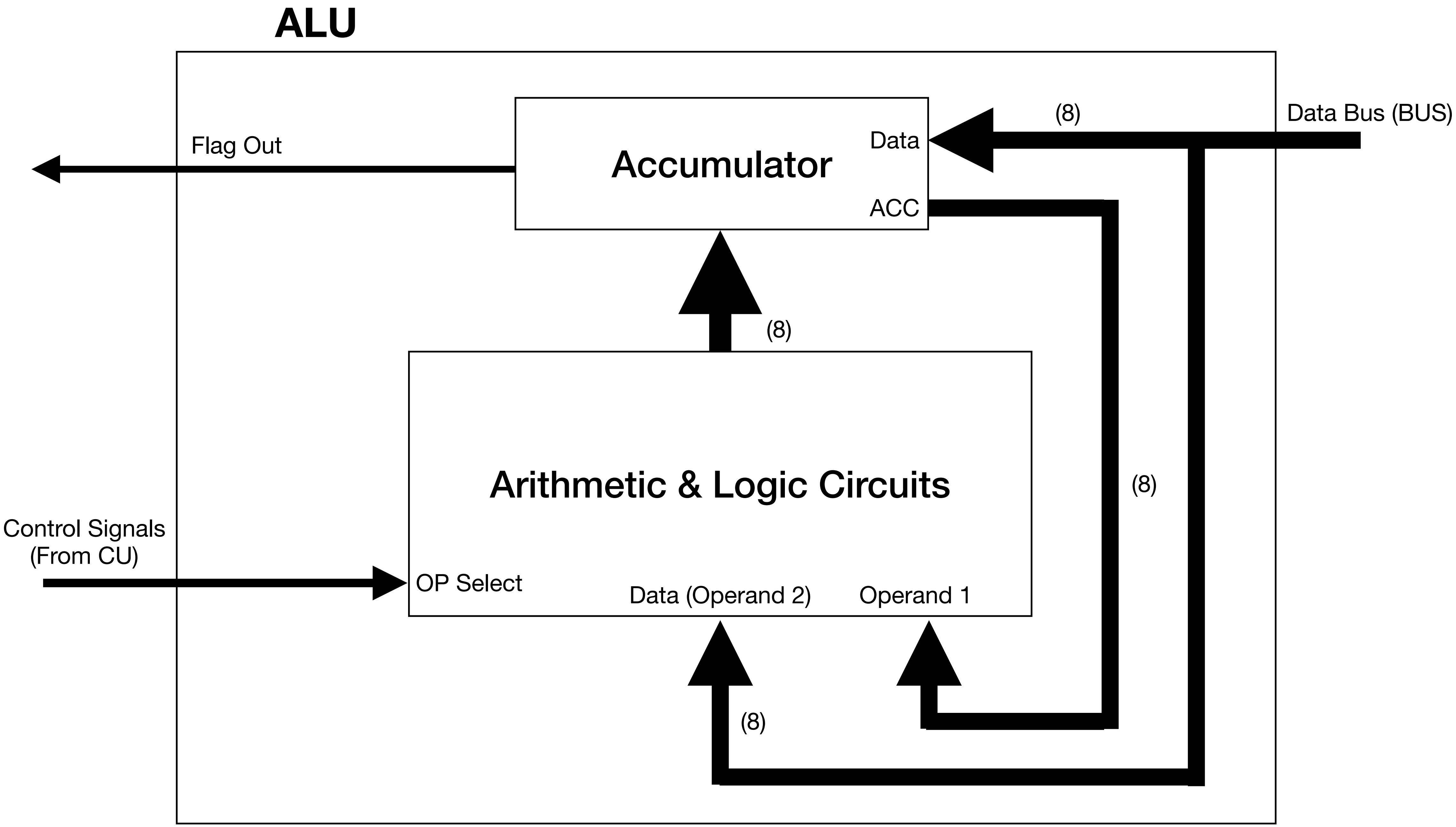

Figure 2. The ALU in Detail

ALU

- ALU Operation
  - Control Unit (CU) sets Operand 2 by loading data in MBR to BUS
  - Operation is done by the ALU, stores result in ACC
  - The number of cycles required by the ALU to finish depends on the instruction (MUL which requires 8 cycles)
  - Since the ACC is tied to the Operand 1 of ALU, the latter can operate directly on the previous result of arithmetic or logical operation
  - Flags: Zero Flag (ZF), Signed Flag (SF), Overflow Flag (OF) and Carry Flag (CF)

ALU

7	6	5	4	3	2	1	0
OF	**-**	**-**	**-**	**-**	**SF**	**CF**	**ZF**
FLAGS Register							

Table 1. Detail of the FLAGS register

ALU

- Instructions (arithmetic)
  - ADD - add ACC with data in BUS, result stored to ACC
  - SUB - sub ACC with data in BUS, result stored to ACC
  - MUL - multiply ACC with data in BUS, result stored to ACC

ALU

- Instructions (logical)
  - AND - **and** ACC with data in BUS, result stored to ACC
  - OR - **or** ACC with data in BUS, result stored to ACC
  - NOT - invert data on ACC, result stored to ACC
  - XOR - **xor** ACC with data in BUS, result stored to ACC
  - SHR - shift ACC 1-bit to the right, result stored to ACC
  - SHL - shift ACC 1-bit to the left, result stored to ACC

ALU

- Instructions (compare* & branch**)
  - BRE - compare ACC and BUS (ACC <- ACC - BUS), set ZF
  - BRNE - compare ACC and BUS (ACC <- ACC - BUS), set ZF
  - BRGT - compare ACC and BUS (ACC <- ACC - BUS), set SF
  - BRLT - compare ACC and BUS (ACC <- ACC - BUS), set SF

* compare operation circuitry is in ALU

** branch operation circuitry is in CU depending on the value of ZF or SF

Control Unit

- Performs instruction fetch, 2-cycle since memory is only 8 bits and instruction is 16 bits (Big Endian)
- Increments PC every fetch (upper and lower byte)
- Decodes instruction fetched by separating instruction code (opcode) and operand
- Executes instruction (data movement operations are handled inside the CU and the rest at the ALU)
- Sets control signals to ALU, data memory and I/O memory

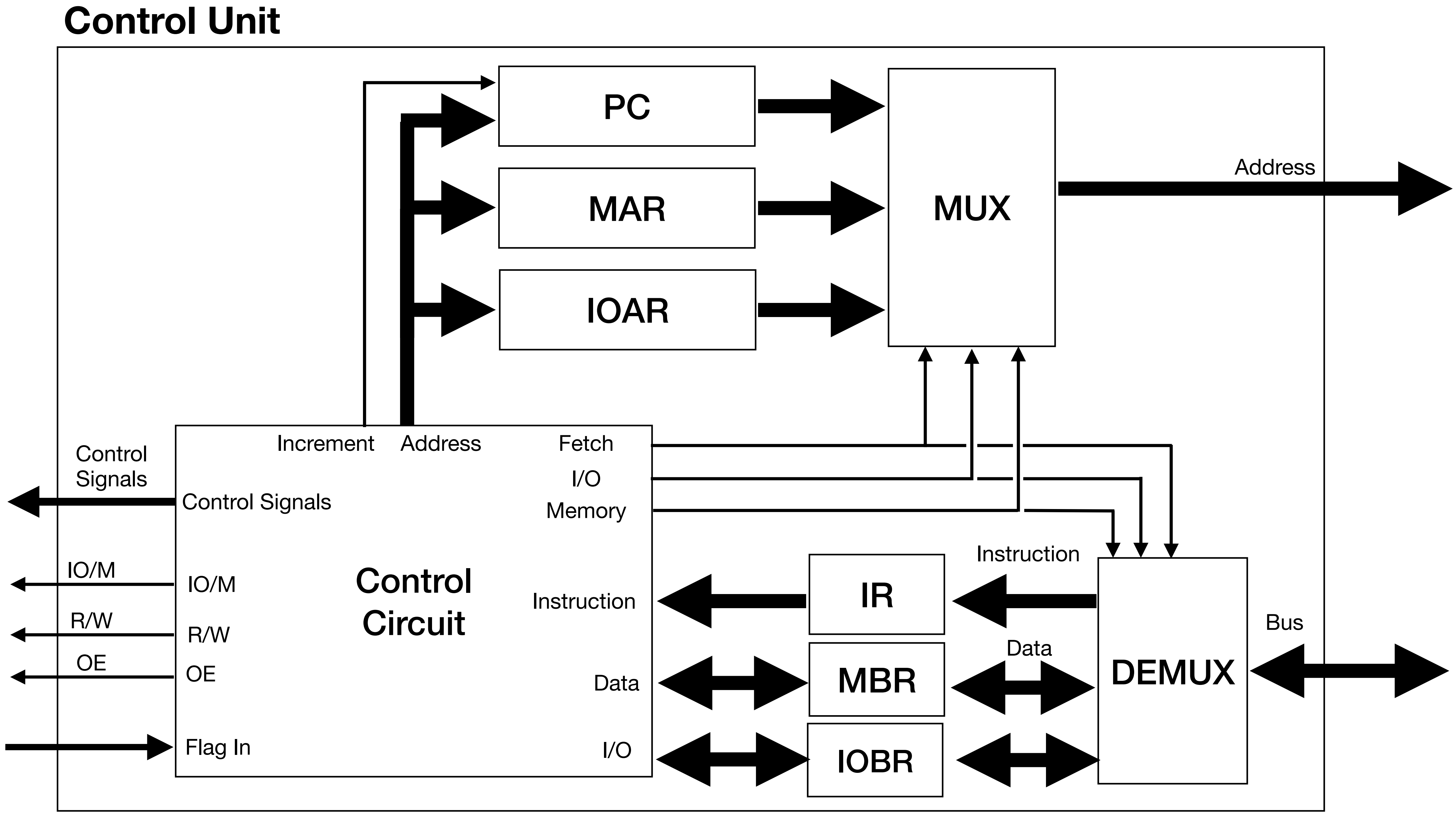

Figure 3. Control Unit Architecture

Control Unit

- Registers:
  - **PC** (Program Counter) - pointer to the next instruction to be fetched, 12 bits
  - **IR** (Instruction Register) - holds the instruction fetched, 16 bits
  - **MAR** (Memory Address Register) - holds the address of the data memory, 11 bits
  - **IOAR** (I/O Address Register) - holds the address of the I/O memory, 11 bits
  - **MBR** (Memory Buffer Register) - holds the data to be written to memory, holds also data read from Main Memory
  - **IOBR** (I/O Memory Buffer Register) - holds the data to be written to I/O memory, holds also data read from I/O Memory

15				11											0
Instruction Code					**Operand**										

(a)

WM (write to memory), where ‘x’ represents the 8-bit literal value															
0	0	0	0	1	0	0	0	x	x	x	x	x	x	x	x
Instruction Code					**Operand**										

(b)

Table 2. Instruction Format: a) format b) example

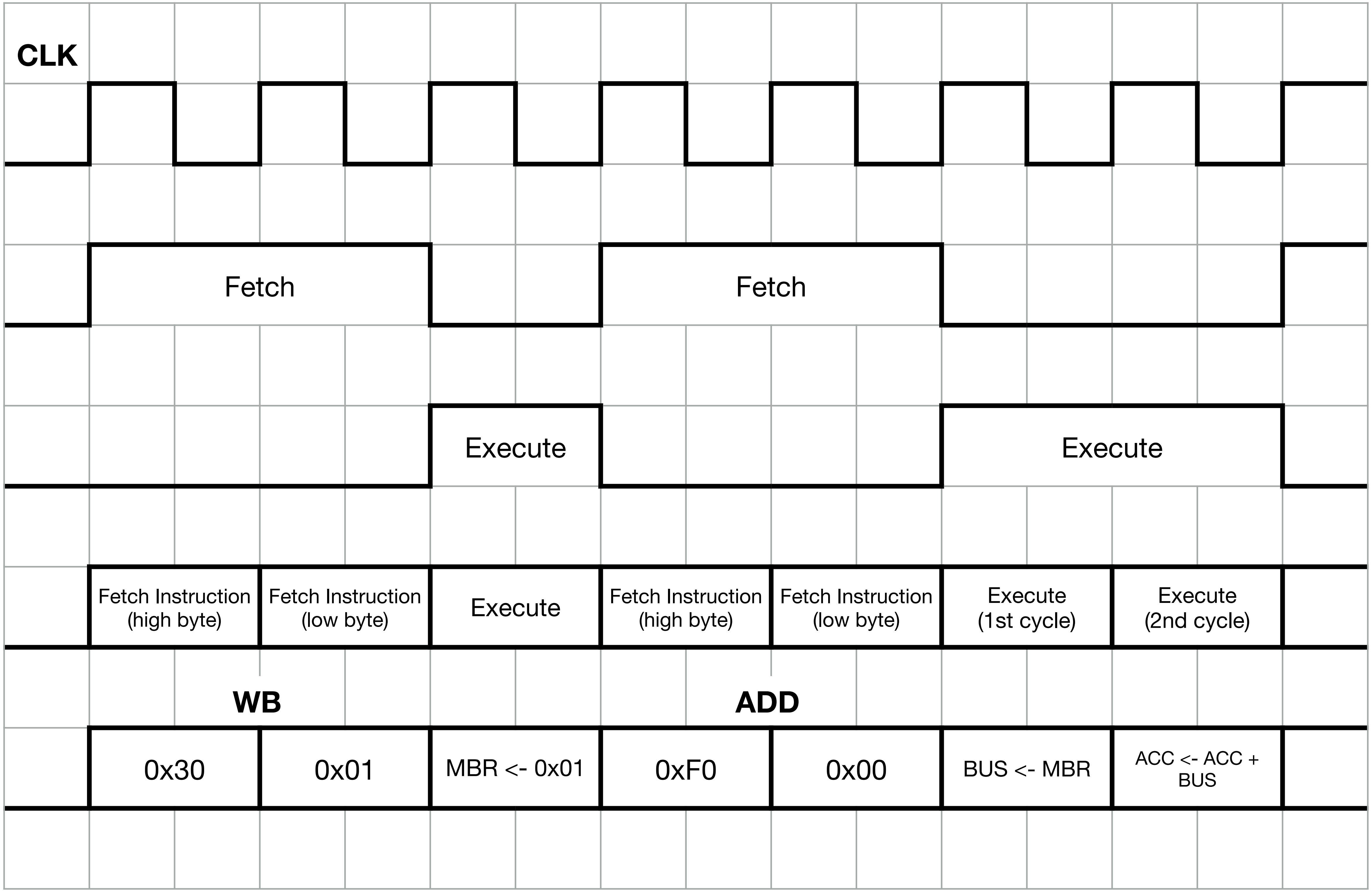

Figure 4. Instruction Cycle Timing Diagram

Control Unit

- Internal Control Signals
  - *Fetch* - indicates an instruction fetch operation (‘1’-fetch, ‘0’-other)
  - *Memory* - indicates a main memory access (‘1’-main memory, ‘0’-other)
  - *IO* - indicates an I/O memory access (‘1’-main memory, ‘0’-other)
- External Control Signals
  - CONTROL - see Table 2 for details
  - IOM
  - RW

<table>
<tr><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="5"><b>Instruction Code</b></td></tr>
</table>

<table>
<tr><td rowspan="2"><b>Instruction Code (inst code)</b></td><td rowspan="2">5-bit instruction code</td><td>valid codes: 0x01 - 0xFF</td></tr>
<tr><td>reserved: 0x00</td></tr>
</table>

Table 3. Control Signals Description

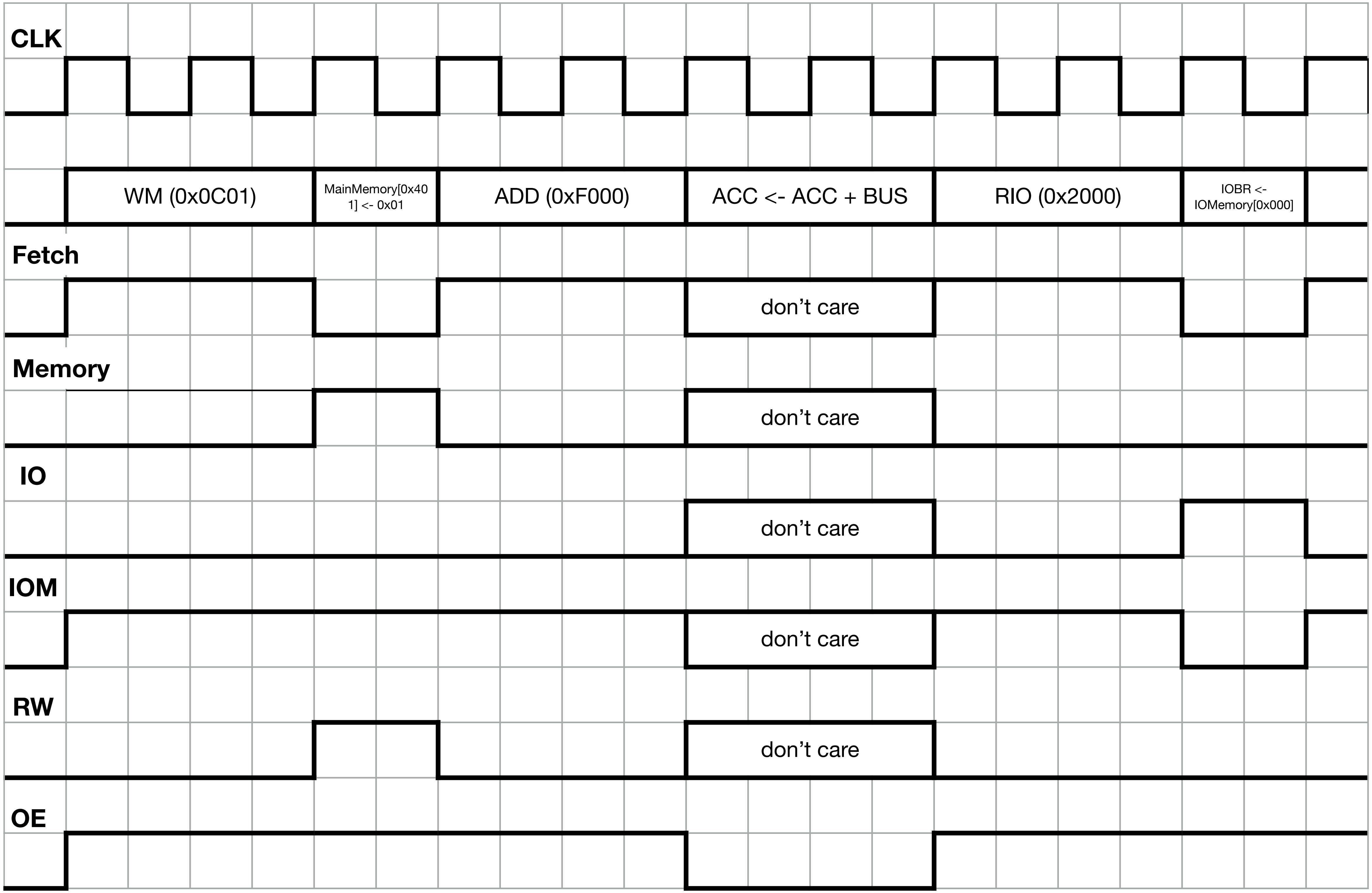

Figure 5. Control Signals Timing Diagram (Synchronous)

Control Unit

- Instructions (data movement, memory/IO)
  - WM (write to memory) - write the 8-bit data on the MBR to memory address pointed to by the MAR (Main Memory)
  - RM (read from memory) - read data from memory address pointed to by the MAR, data stored to MBR (Main Memory)
  - WIO (write to IO) - write the 8-bit data on the IOBR to the memory address pointed to by the MAR (IO Memory)
  - RIO (read from IO) - read data from memory address pointed to by the IOAR, data stored to IOBR (IO Memory)

Control Unit

- Instructions (data movement, register)
  - WB (write to memory buffer) - write literal 8-bit value on the MBR
  - WIB (write to IO buffer) - write literal 8-bit value on the IOBR
  - SWAP (swap MBR & IOBR) - swap the data in MBR to IOBR vice-versa
- Instructions (program control)
  - BR (branch) - program jumps to program memory address specified
  - EOP (end of program) - no more instruction after EOP, CU will halt

Control Unit

- Instructions (read/write data from/to ACC)
  - WACC (write to ACC) - write an 8-bit data on BUS to ACC
  - RACC (read from ACC) - read ACC data and load to BUS

Control Unit

- Instructions (compare* & branch**)
  - BRE - if ZF=1, branch to specified *program memory* address
  - BRNE - if ZF=0, branch to specified *program memory* address
  - BRGT - if SF=0, branch to specified *program memory* address
  - BRLT - if SF=1, branch to specified *program memory* address

* compare operation circuitry is in ALU

** branch operation circuitry is in CU depending on the value of ZF or SF

Main Memory

- 8-bit x 2048
- Program memory inclusive address: 0x000 - 0x3FF
- Data memory inclusive address: 0x400 - 0x7FF
- Read/write operation via **RW**
- Main Memory access via **IOM**
- Output enable or disable via **OE**

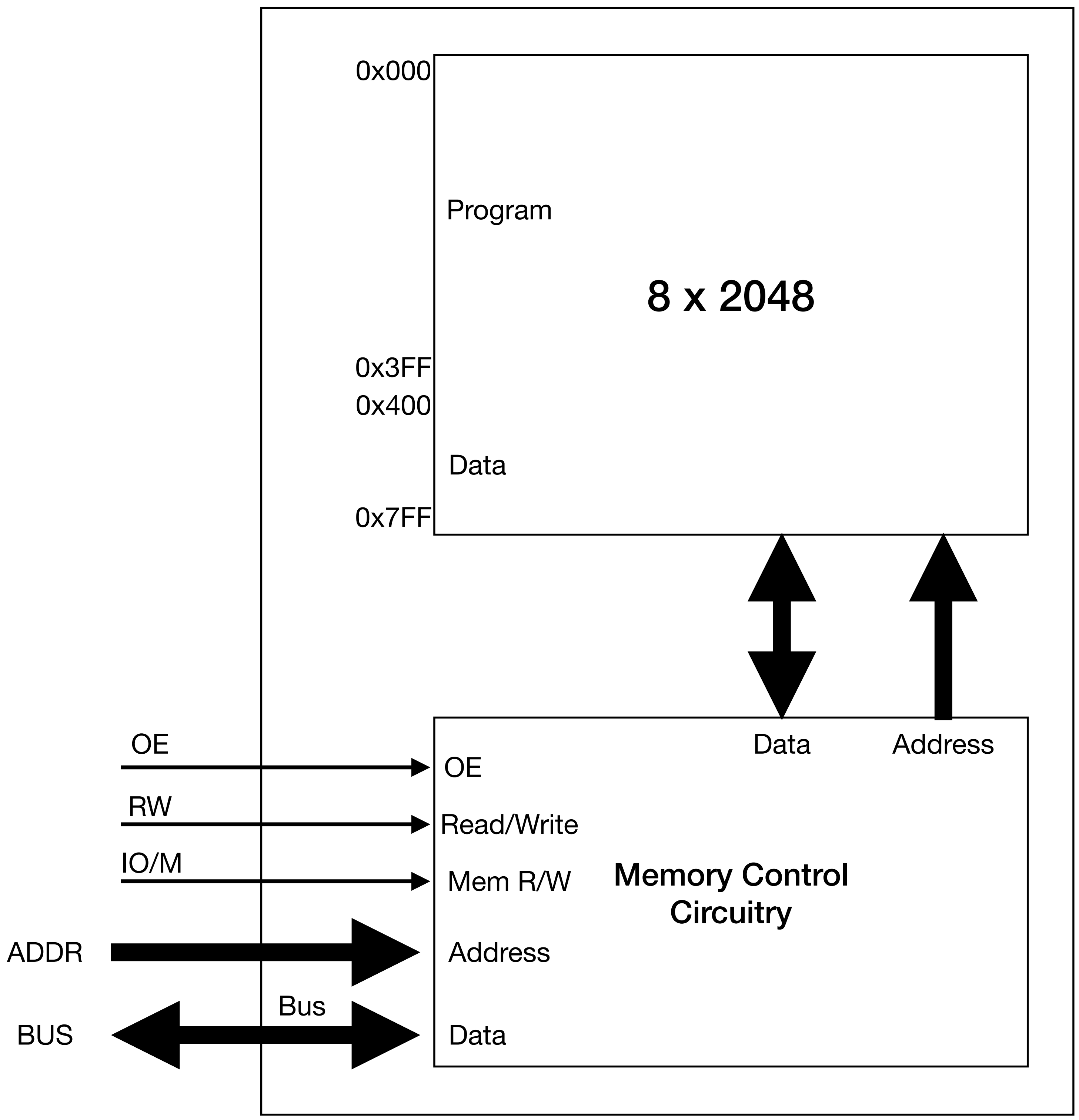

Figure 6. Main Memory

Main Memory Organization

- Composed of sixteen (16) 32x32 bit chips for total of 16Kbit
- Two (2) chip groups A & B
- Organized into (32 + 32) bit columns and (32 x 8) rows
- Chip Select (cs) identify the chip group A or B
- Address bus is composed of *column address*, *row address* and *chip group selector*

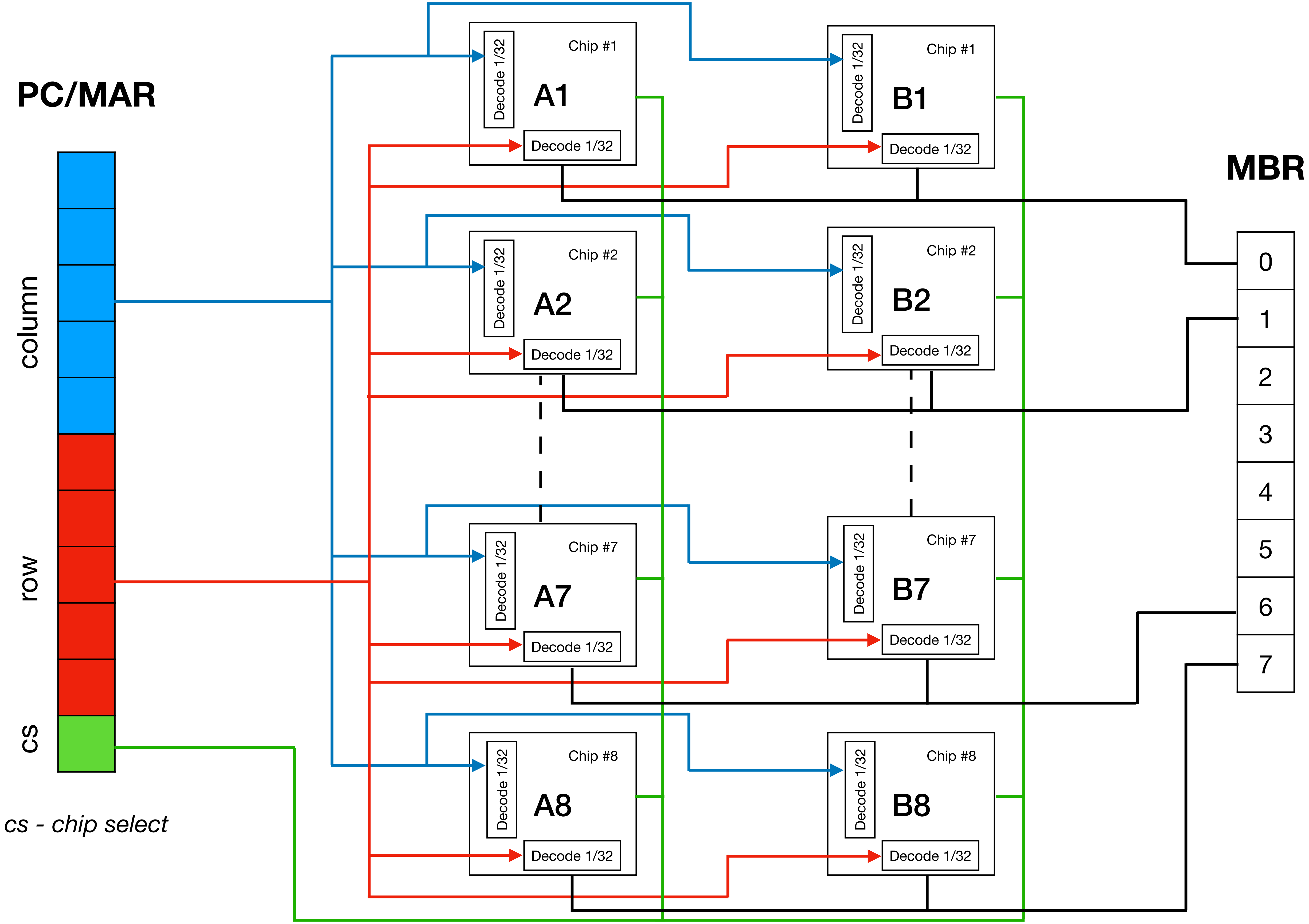

Figure 7. Main Memory Organization

Address Decoding

- Column is decoded and the bit is stored/read from the column address
- Row address is decoded and the bit is stored/read from the row address
- Therefore bit 0 of the MBR (BUS) is stored on *(col, row)* of chip A1 if cs = 0 or B1 if cs = 1
    - bit x -> (col, row) of A(x+1) if cs = 0
    - bit x -> (col, row) of B(x+1) if cs =1

I/O Memory

- Isolated from Main Memory (as I/O Memory)
- Maximum of 32 addressable latch/buffer
- IO address decoding, enables the proper latch or buffer
- latches for output signals and buffer for input signals
- I/O read and write control via IOM and RW control lines
- Data latched (output) when IOM=0 and RW=1
- Data buffered (input) when IOM=0 and RW=0

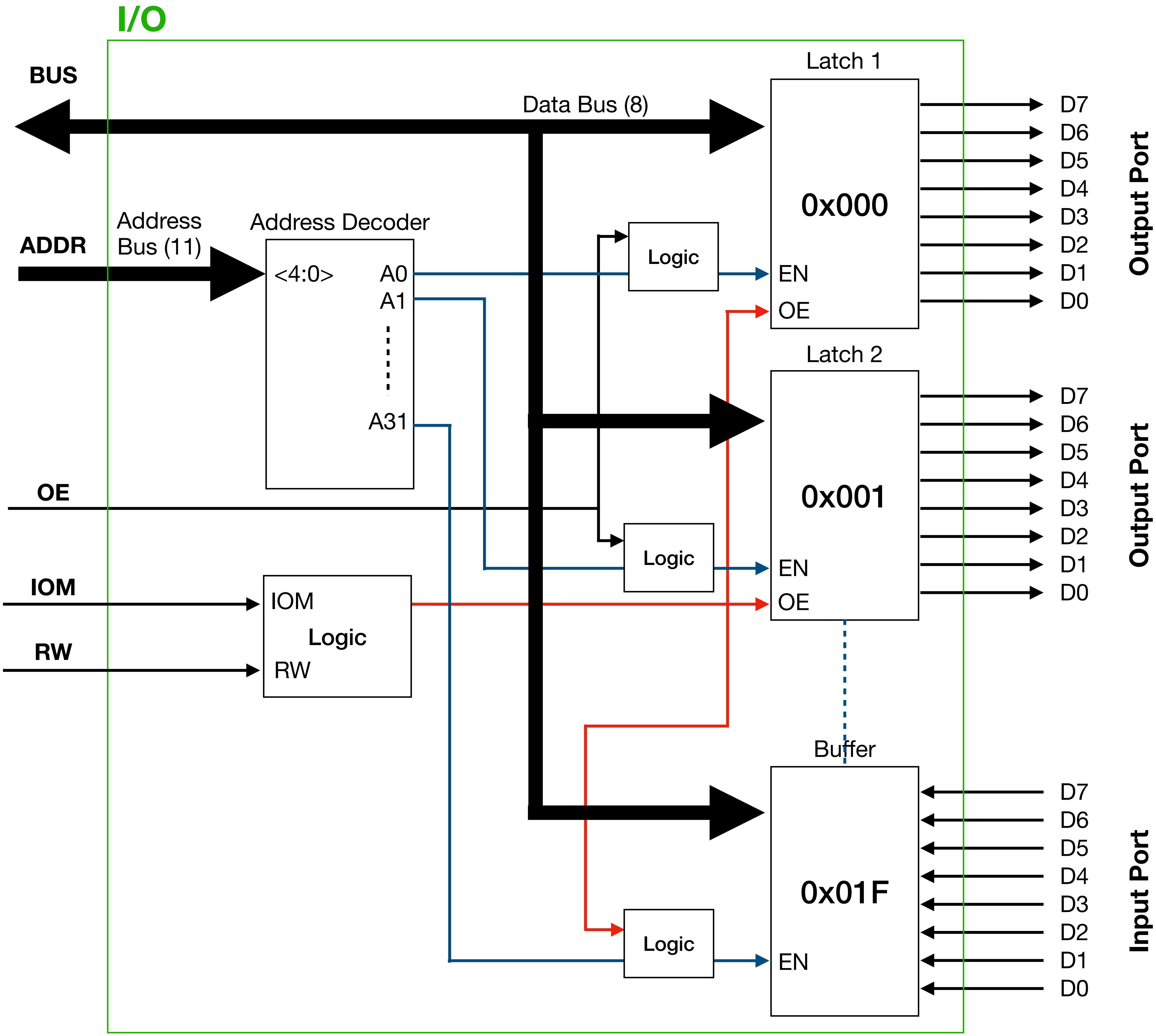

Figure 8. I/O Memory

I/O Memory

- Latch (output) address: 0x000 to 0x00F
- Buffer (input) address: 0x010 to 0x01F
- Up to 128 output ports in 16 groups of 8-bits
- Up to 128 input ports in 16 groups of 8-bits

Instruction Set

Table 3a. Instruction Set (Arithmetic & Logical)

Semantics	Operation Description	Syntax	Operation	Flags	Inst Code	Instruction Format	Remarks
	Arithmetic & Logical						
ADD	Adds the data on the BUS to ACC register, sum stored to ACC	ADD	ACC <- ACC + BUS	ZF, SF OF, CF	11110	1111 0uuu uuuu uuuu	u - unused bits
SUB	Subtract the data on the BUS from the ACC register, difference stored to ACC	SUB	ACC <- ACC - BUS	ZF, SF OF, CF	11101	1110 1uuu uuuu uuuu	u - unused bits
MUL	Multiply the value of ACC to BUS, product stored to ACC	MUL	ACC <- ACC x BUS	ZF, SF OF, CF	11011	1101 1uuu uuuu uuuu	u - unused bits
AND	AND the value of ACC and BUS, result stored to ACC	AND	ACC <- ACC & BUS	ZF	11010	1101 0uuu uuuu uuuu	u - unused bits
OR	OR the value of ACC and BUS, result stored to ACC	OR	ACC <- ACC | BUS	ZF	11001	1100 1uuu uuuu uuuu	u - unused bits
NOT	Complement the value of ACC, result stored to ACC	NOT	ACC <- !ACC	ZF	11000	1100 0uuu uuuu uuuu	u - unused bits
XOR	XOR the value of ACC and BUS, result stored to ACC	XOR	ACC <- ACC ^ BUS	ZF	10111	1011 1uuu uuuu uuuu	u - unused bits
SHL	Shift the value of ACC 1 bit to the left, CF will receive MSB of ACC	SHL	ACC <- ACC << 1, CF <- ACC <7>	CF	10110	1011 0uuu uuuu uuuu	u - unused bits
SHR	Shift the value of ACC 1 bit to the right, CF will receive LSB of ACC	SHR	ACC <- ACC >> 1, CF <- ACC <0>	CF	10101	1010 1uuu uuuu uuuu	u - unused bits

Instruction Set

Table 3b. Instruction Set (Data Movement)

Semantics	Operation Description	Syntax	Operation	Flags	Inst Code	Instruction Format	Remarks
	Data Movement						
WM	Write data in MBR to memory at address pointed to by MAR	WM addr	BUS <- MBR	NA	00001	0000 1xxx xxxx xxxx	x - address
RM	Read data from memory with the specified address, stores data to MBR	RM addr	MBR <- BUS	NA	00010	0001 0xxx xxxx xxxx	x - address
RIO	Read data from IO memory with the specified address, stores data to IOBR	WIO addr	IOBR <- BUS	NA	00100	0010 0xxx xxxx xxxx	x - address
WIO	Write data in IOBR to memory at address pointed to by IOAR	WIB addr	BUS <- IOBR	NA	00101	0010 1xxx xxxx xxxx	x - address
WB	Write literal value to MBR	MBR l	MBR <- literal	NA	00110	0011 0000 xxxx xxxx	x - literal
WIB	Write literal value to IOBR	WIB l	IOBR <- literal	NA	00111	0011 1000 xxxx xxxx	x - literal
WACC	Write data on BUS to ACC	WACC	ACC <- BUS	NA	01001	0100 1uuu uuuu uuuu	u - unused bits
RACC	Move ACC data to BUS	RACC	BUS <- ACC	NA	01011	0101 1uuu uuuu uuuu	u - unused bits
SWAP	Swap data of MBR and IOBR	SWAP	MBR <- IOBR, IOBR <- MBR	NA	01110	0111 0uuu uuuu uuuu	u - unused bits

Instruction Set

Table 3c. Instruction Set (Program Control)

Semantics	Operation Description	Syntax	Operation	Flags	Inst Code	Instruction Format	Remarks
	Program Control						
BR	Branch to specified address	BR addr	PC <- addr	NA	000112	0001 1xxx xxxx xxxx2	x - address
BRE	Compare ACC and BUS, if equal branch to address specified	BRE addr	ACC <- ACC - BUS if ZF = 1 then PC <- addr	ZF, SF OF, CF	101002	1010 0xxx xxxx xxxx2	x - address
BRNE	Compare ACC and BUS, if not equal branch to address specified	BRNE addr	ACC <- ACC - BUS if ZF = 0 then PC <- addr	ZF, SF OF, CF	100112	1001 1xxx xxxx xxxx2	x - address
BRGT	Compare ACC and BUS, if ACC > BUS, branch to address specified	BRGT addr	ACC <- ACC - BUS if SF = 0 then PC <- addr	ZF, SF OF, CF	100102	1001 0xxx xxxx xxxx2	x - address
BRLT	Compare ACC and BUS, if ACC < BUS, branch to address specified	BRLT addr	ACC <- ACC - BUS if SF = 1 then PC <- addr	ZF, SF OF, CF	100012	1000 1xxx xxxx xxxx2	x - address
EOP	End of program, no more instructions, CU will halt	EOP	NA	NA	111112	1111 1uuu uuuu uuuu2	u - unused bits

References

- Stallings, W. Computer Organization and Architecture, 6th edition, Pearson Education, Inc. (2003).
- Brey, B. The Intel Microprocessors: architecture, programming and interfacing, 8th edition, Pearson Education, Inc. (2003).
- PIC16XXX Architecture, PIC16F84A Data Sheet, Microchip Technology Inc. (2001).

Acknowledgment

- Transition Semester students of CpE 415N for being the testers that emulated and evaluated the architecture.

Document Version

Date	Version	Author	Changes
May 2018	1.0	Van B. Patiluna	Initial draft.
June 2018	2.0	Van B. Patiluna	- Revised architecture block diagram - Added memory orgnization - Revised control unit architecture
July 2018	2.1	Van B. Patiluna	- Added SWAP instruction - Finalized instruction set
September 2019	2.2	Van B. Patiluna	- Removed the DIV, WACCH, WACCL, RACCH, RACCL instructions - Added WACC and RACC instructions - Changed ACC width to 8 bits - Revised Figure 8 - Improvements on the instruction set table
February 28, 2021	2.3	Van B. Patiluna	- Added the FLAGS register in the ALU - Added the Table 1 (FLAGS register details) - Rearranged the order of the slides
February 28, 2021	2.4	Van B. Patiluna	- Added Table 2 (Instruction Format) - Corrected some factual errors - Updated Table 3c - Corrected Figure 5