IGEE University M'Hamed Bougara of Boumerdès	Last Name: ..... First Name: ..... Group:	Grade :

EE321: Computer Architecture
Final Exam Solution (1h30)
2019 – 2020

Notes: Answer briefly and clearly using the provided space. No extra sheet will be accepted.

1. Control Unit (9 pts)

Consider a simple CPU with a single accumulator AC given in Figure 1. All the arithmetic and logic instructions use the accumulator AC a source and as a destination register. The micro-operations of the fetch cycle, interrupt cycle, and execute cycle of the ADD instruction are given in Table 1.

1.1 Based on the given architecture, which registers are used for logic and arithmetic operations. Justify.

AC and MBR registers. They are connected to the ALU through the control signal C_7 and C_6. *(1pt)*

1.2 Why the first micro-operation of the execute cycle of ADD instruction is MAR←MBR(X)?

ADI instruction is using the direct mode and the operand of the instruction IR(X) is the address of the data, so we should have MAR ← IR(X). At the end of the fetch cycle, we have MBR=IR, so MAR ← IR(X) is the same as MAR ← MBR(X). *(1 pt)*

1.3 Give the micro-operations of the execute cycle of the instruction ADI (see Table 2).

T1: AC ← AC+MBR(X). *(1 pt)*

1.4 Give the micro-operations of the indirect cycle and execute cycle of the ADN instruction.

Indirect cycle:
T1: MAR ← MBR(X)
T2: MBR ← Memory
Execute cycle:
T1: MAR ← MBR
T2: MBR ← Memory
T3: AC ← AC+MBR *(1 pt)*

1.5 We add a stack memory pointer register SP to the architecture. Give the micro-operations of execute cycle of the PUSH and POP instruction (see Table 2).

Execute cycle of PUSH:
T1: MAR ← SP
MBR ← AC
T2: Memory ← MBR
SP ← SP +1
Execute cycle of POP:
T1: SP ← SP -1
MAR ← SP
T2: MBR ← Memory
T3: AC ← MBR *(1 pt)*

1.6 Based on the micro-operations of the interrupt cycle, which type (vectored or non-vectored) of interrupt is implemented? Justify.

We have: MBR ← Device *and* PC ← MBR
Non-Vectored interrupt, because the address of the ISR is provided by the device requesting an interrupt. *(1 pt)*

1.7 Why in Table 3, we do not have a control signal for ADN instruction?

Because the execute cycle of indirect ADN instruction is the same as the execute cycle of the direct ADD instruction, so we use ADD control signal for ADN instruction. (1 pt)

1.8 Considering only the instructions given in Table 2 and using the control signals given in Table 3, give the boolean expression of C_8 controling the data transfer from MBR to MAR.

C_8 is active in indirect cycle (at T1) and execute cycle for ADD instruction (at T1):
$==> C_8 = N \cdot T_1 + E \cdot ADD \cdot T_1$ *(1 pt)*

1.9 We assume the data transfer from SP to MAR register is controled by the control signal C_{15}. Using the control signals given in Table 3, give the boolean expression of C_{15}.

C_{15} is active in PUSH and POP execute cycle (at T1) and interrupt cycle (at T1)
$==> C_{15} = E \cdot T_1(POP + PUSH) + I \cdot T_1$ *(1 pt)*

2. Instruction Set and Addressing modes (5 pts)

Consider a simple 8-bit CPU with the following specifications: 8-bit address bus, memory is byte addressable, opcodes and addressing mode codes are given in Table 4, binary codes of the general purpose registers (GPR) are given in Table 5, and the instructions format is given in Table 6.

2.1 Give the instructions that are using a direct access to the memory.

AD Rd, [X]
AD Rd,# Rs
AD Rd, [Rs]
LD Rd, Rs
LD Rd, [X]
ST Rd, Rs
LD Rd, # Rs *(1 pt)*

2.2 Which instructions are using a displacement mode? Give the implicit and explicit operand.

AD Rd, # Rs (implicit operand:R0, explicit operand: Rs).
LD Rd, # Rs (implicit operand:R0, explicit operand: Rs).
JP X (implicit operand: PC, explicit operand: X).

2.3 For the immediate mode, the size of the X operand is 6 bits, explain how we can load 8-bit value into a register.

We use two instructions: ML and MH. *(1 pt)*

2.4 Give the HEX code of the instructions: a) `ML R3,08H`, b) `MH R3,0FH`, and c) `JP -7`.

a) The binary code of the instruction ML R3,08H is: 0001 0000 1100 1000 = 10C8H
b) The binary code of the instruction MH R3,0FH is: 0110 0000 1100 1111 = 60CFH
c) The binary code of the instruction JR -7 is: 0101 XXXX XX11 1001 = 5039H

2.5 Write an assembly progam to save the value 84H into memory location F6H.

ML R1, 04H
MH R1, 08H
ML R2, 06H
MH R2, 0FH
ST R2, R1
HL *(1 pt)*

3. Memory and Basic Concepts (6 pts)

3.1 Consider à 64-bit wide 2GBytes Double Data Rate 3 (DDR3 SDRAM) DIMM memory and assuming a bus clock frequency of 400MHz, give in Mbits/s, the trasfer rate of the memory.

*Transfer rate = 400*2*64 = 51200 Mbits/s =51.2 Tbits/s*

3.2 How many 512Mx8-bit memory chips are needed to design the memory of question 3.1.

The memory module data width is 64 bits, so we need at least, 8 memory chips.
The memory module depth is 256 Mlocations, so we need 8 memory chips. *(1 pt)*

3.3 Consider the 2's complement representation using 16 bits, what is the numbers of values, the smallest and the greatest value that can be represented (give the results in decimal)?

Number of values: $2^{16} = 65536$.
Greatest value: +32367 ; Smallest value: -32768 *(0.5pt)*

3.4 Consider the 2's complement representation using 16 bits, give the 2's complement representation of the greatest and smallest value.

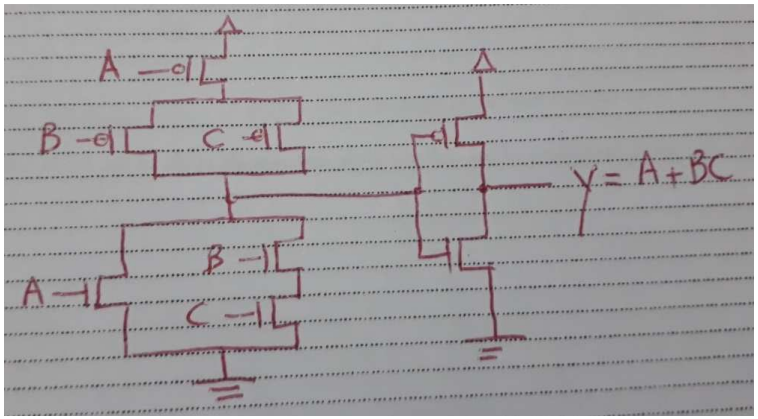
$(+32\ 367)_{10} = (0111\ 1111\ 1111\ 1111)_{2's\ C}$
$(-32\ 768)_{10} = (1000\ 0000\ 0000\ 0000)_{2's\ C}$ *(0.5pt)*

3.5 Which types of trasnfer-of-control operations are implemented in Z80 processor?

1) Branch (jump) operations
2) Procedure call operations

3.6 Draw the CMOS schematic for the gate that implements the function: $Y = A + B(CD + C)$.

$Y = A + B(CD + C) = A + BC = \overline{\overline{A + BC}}$ *(2 pts)*



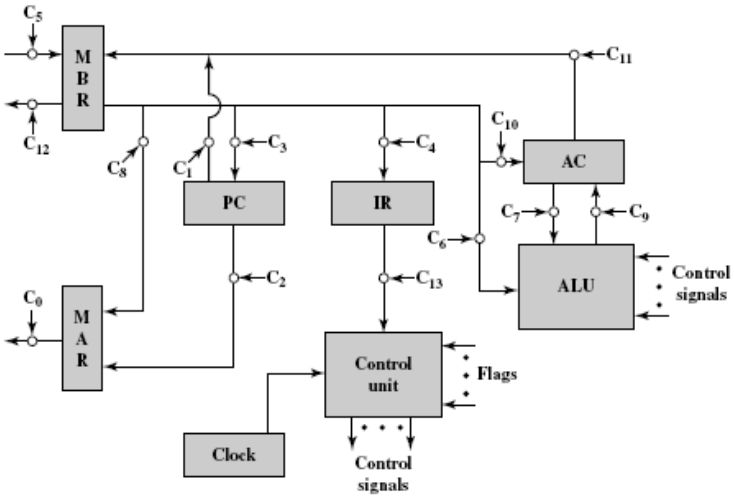


Figure 1: Data path and control signals [© Stallings].

Clock cycle	Fetch cycle	Interrupt cycle	Execute cycle of ADD instruction (i.e. ADD AC, X)
T1	*MAR ← PC*	*MBR ← PC* *MAR ← SP*	*MAR ← MBR(X)*
T2	*MBR ← Memory* *PC ← PC+1*	*Memory ← MBR* *SP ← SP+1*	*MBR ← Memory*
T3	*IR ← MBR*	*MBR ← Device*	*AC ← AC+MBR*
T4		*PC ← MBR*	
T5			

Table 1: Micro-opérations of fetch cycle, interrupt cycle, and execute cycle of the ADD instruction.

Instruction	Description
ADI AC, X	AC = AC + X
ADD AC, X	AC = AC + [X]
ADN AC, X	AC = AC +[[X]]
PUSH AC	[SP] = AC ; SP = SP +1
POP AC	SP = SP-1 ; AC = [SP]

Table 2: Instructions description.

Signal	Description
I	I=1 for interrupt cycle
F	F=1 for fetch cycle
E	E=1 for execute cycle
N	N=1 for indirect cycle
ADI	ADI=1 for ADI operation
ADD	ADD=1 for ADD operation
POP	POP=1 for pop operation
PUSH	PUSH=1 for push operation

Table 3: Signals used in the control unit.

Instruction	Opcode (hex)	Description	Addressing mode code (binary)
ML Rd, X	1H	$Rd_{3:0} = X_{3:0}$	000
MH Rd, X	6H	$Rd_{7:4} = X_{3:0}$	000
MV Rd, Rs	1H	Rd = Rs	001
LD Rd, [X]	2H	Rd = [X]	011
LD Rd, # Rs	2H	Rd = [R0+Rs]	101
LD Rd, Rs	2H	Rd = [Rs]	010
ST Rd, Rs	3H	[Rd] = Rs	010
AD Rd,X	4H	Rd = Rd+X	000
AD Rd, Rs	4H	Rd = Rd+Rs	001
AD Rd, [Rs]	4H	Rd = Rd+[Rs]	010
AD Rd, [X]	4H	Rd =Rd+ [X]	011
AD Rd,[[X]]	4H	Rd=Rd+[[X]]	100
AD Rd,# Rs	4H	Rd=Rd+[R0+ Rs]	101
JP X	5H	PC = PC+X	/
HL	FH	Stop the execution of the program	/

Rd = Source/destination register (it can be any GPR).
Rs = Source register (it can be any GPR)
R0 = Register R0.
[X] = content of memory location X.

Table 4: Instructions description.

Register	R0	R1	R2	R3	R4	R5	R6	R7
Code	000	001	010	011	100	101	110	111

Table 5: Binary codes of GPRs.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Opcode				Addressing mode			Destination register			Source operand					

Source operand can be register code, address, or value. If register code, bits 3 to 5 are equal to 0.
Destination register contains the binary code of source/destination register.

Table 6: Instructions format.