CSE Department – Faculty of Engineering - MSA
Spring 2025
CSE145 Computer Architecture
Project Report
Course Instructor Dr. Manal Mostafa
TA. Eng Yomna Sameer

Name	ID
Youssef Mohamed Fathi	247349
Baraa yousef	241715
mohamed walid	245865
Omar khaled	247497
Mohamed wessam	246423
Ramez Rafat Zaky	247585

Introduction:

The simple computer we designed is made out of different modules which are:

- Processor registers
- Control unit (hardwired)
- Bus multiplexer
- RAM
- EEPROM

Every part of the computer is synchronized with the same clock. Furthermore, the control unit fetches the binary code and the executes the necessary control signal for the specified operation code.

The most important part in this computer in my opinion is the control unit it contains the sequence timer connected to a decoder. Each operation to execute needs to do some microoperation like moving data from a register to another register. Each microoperation takes one clock cycle. In this computer an operation can take up to 16 clock cycles to execute.

How it works:

When the sequencer timer is at T0 the computer first loads PC to AR the at T1 the computer loads word at AR and stores it in IR and increments PC then at T2 the computer stores IR(0-11) to AR and check bit I then at T3 the operation execution begins this was the instruction cycle.

However we made an interrupt cycle also at T0 AR will be loaded with the subroutine address then at T1 TR will be loaded from PC then at T2 M[AR] will be loaded with TR then at T3 PC will be loaded with the address of the interrupt subroutine then at T4 increment PC.

The instruction set:

		Hexadecimal code		
	symbol	I=0	I=1	description
Memory reference	AND	0xxx	8xxx	And memory word to AC
	ADD	1xxx	9xxx	Add memory word to AC
	LDA	2xxx	Axxx	Load memory word to ac
	STA	3xxx	Bxxx	Store AC to memory location
	BUN	4xxx	Cxxx	Branch unconditionally
	BSA	5xxx	Dxxx	Branch save return address
	ISZ	6xxx	Exxx	Increment memory word and skip if zero
Register reference	CLA	7800		Clear AC
	CLE	7400		Clear E
	CMA	7200		Complement AC
	SHR	7080		Shift AC right
	SHL	7040		Shift AC left
	INC	7020		Increment AC
	SPA	7010		Skip next instruction if AC positive
	SNA	7008		Skip next instruction if AC negative
	SZA	7004		Skip next instruction if AC zero
	SZE	7002		Skip next instruction if E zero
	HLT	7001		Halt computer
IO reference	INP	F800		Input Character to AC
	OUT	F400		Output character fro AC
	SKI	F200		Skip on input flag
	ION	F080		Interrupt ON
	IOF	F040		Interrupt OFF

My own implemented instructions To help me further develop my computer	PTA	FFFx	Peripheral on bus to AC where x is the address on bus
	ATP	FFEx	AC to peripheral on bus where x is the address on bus

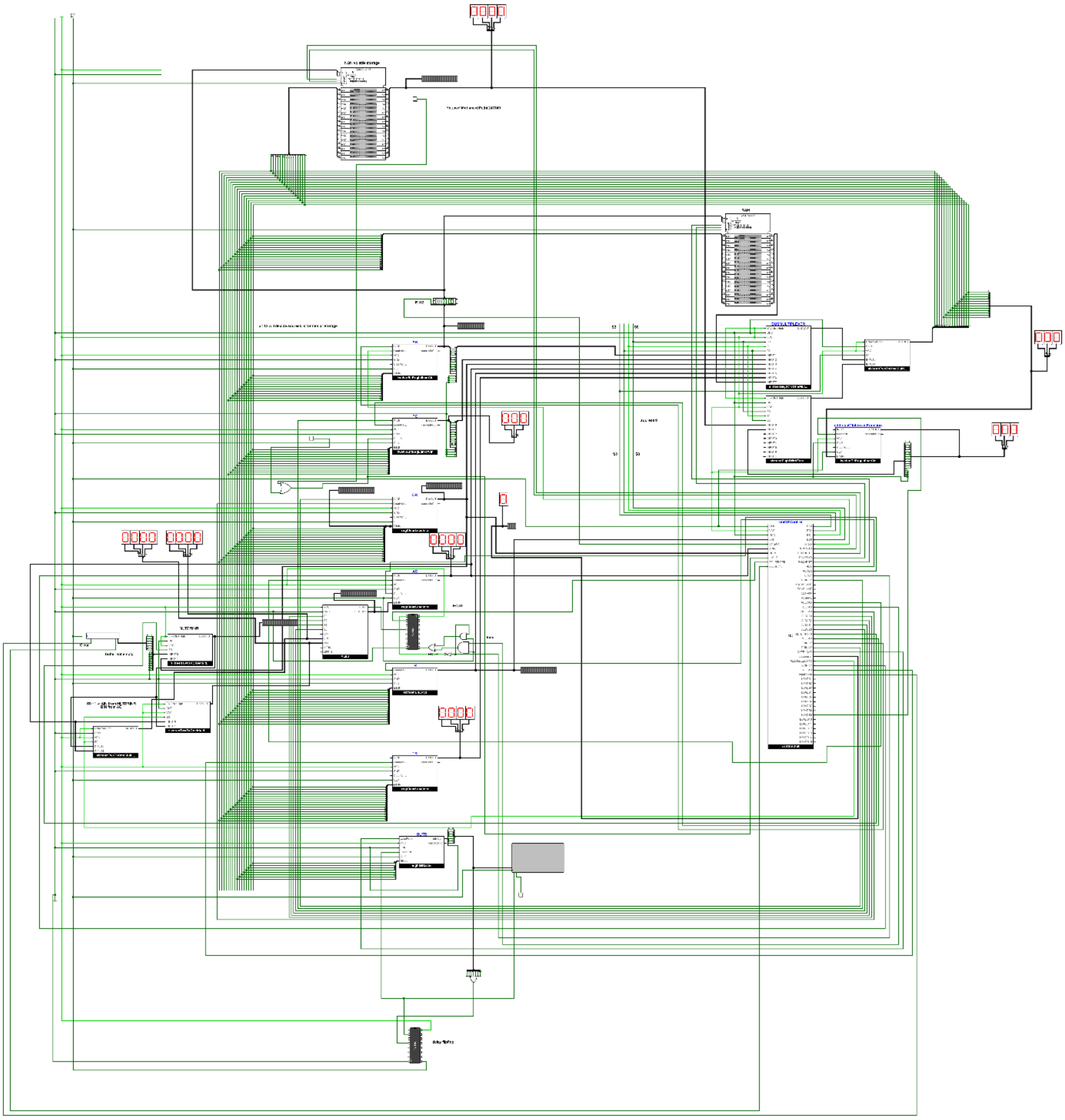
I = 0 for direct address and I = 1 for indirect address.

Processor registers:
AR (12bit): address register.
PC (12bit): program counter.
DR (16bit): data register.
AC (16bit): accumulator.
IR (16bit): instruction register.
TR (16bit): Temporary register used to handle interrupts.
OUTR(8bit): output data register.
INPR(8bit): input data register (can work with interrupt)

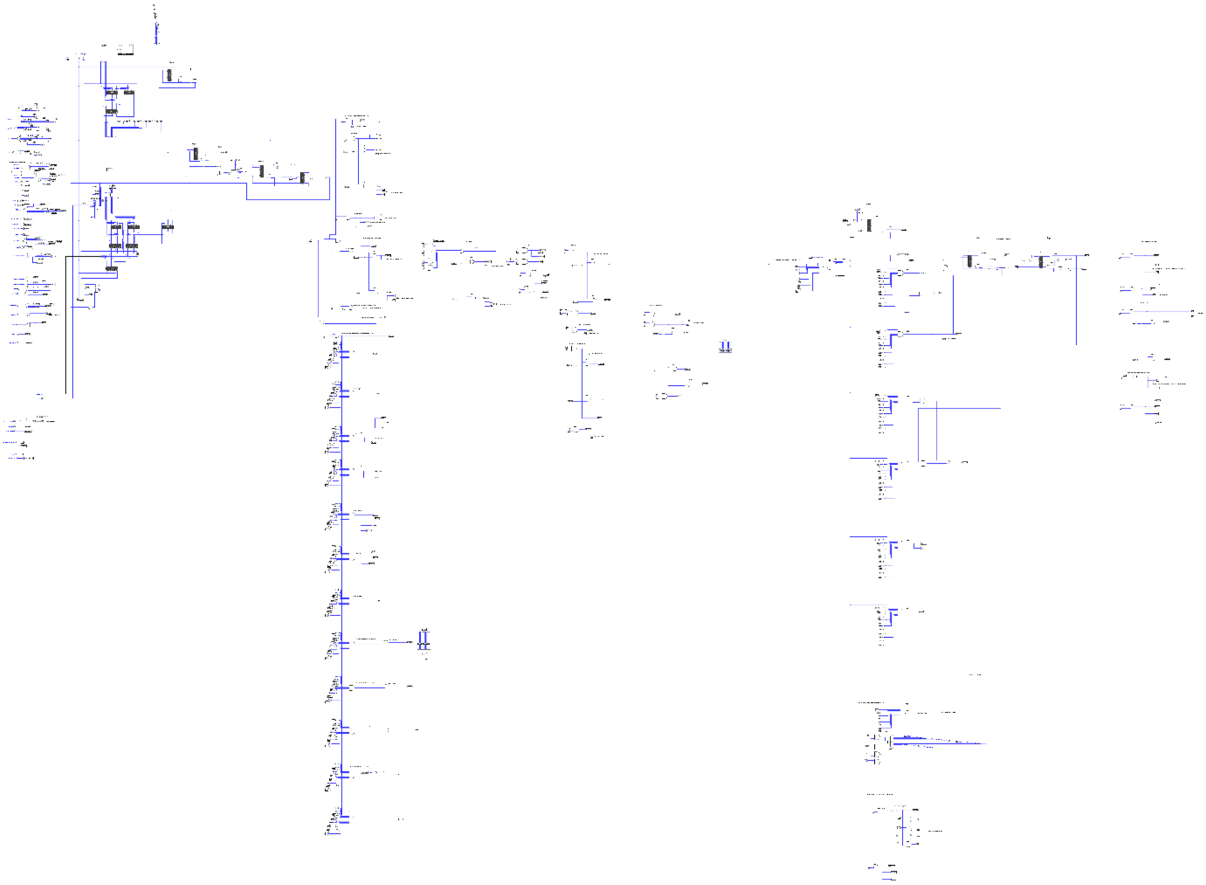
RAM address starts at $(0800)_{16}$
EEPROM address starts at $(0000)_{16}$

So we can write a bootloader to copy data from EEPROM to RAM very easily using just LDA and STA with in a loop and the use BUN to change the PC register to the new address.

Picture of the computer:



Control unit :



A simple program I wrote to test interrupts, EEPROM and RAM addressing

address	data	comment
000	LDA 005	AC < M[005]
001	ATP 9	P9 < AC
002	ION	Enable interupt
003	BUN 000	Jump to address 000 (loop)
004	HLT	Stop computer (will not be executed)
005	0800	Address of interrupt subroutine(in ram)
800	0000	This will contain the address to return to
801	STA 806	Store initial ac value
802	INP	Input character to AC
803	OUT	Write character from ac to terminal screen
804	LDA 806	Load initial ac value
805	BUN 800 I = 1	Jump to address at $(800)_{16}$
806	0000	Place holder for initial ac value

Conclusion:

In summary, designing a simple processor involves a careful balance between efficiency, functionality, and scalability. By focusing on fundamental components such as the control unit, arithmetic logic unit, and memory integration, we can create a processor capable of executing basic instructions effectively. Through thoughtful architecture design and optimization techniques, we enhance processing performance while ensuring ease of implementation. This foundational approach not only provides insight into computer architecture but also serves as a stepping stone for more advanced processor designs in the future.

Reference:

Mano, M. M. (1993). Computer system architecture (3rd ed.). Prentice Hall.