
بسم الله الرحمن الرحيم

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

گزارش پروژه درس معماری کامپیوتر
رشته : مهندسی کامپیوتر
گرایش: سخت افزار

عنوان :

SAYEH Basic Computer

استاد :

دکتر سعید شیری قیداری

تدریس یاران :

فرزان دهباشی – پرهام الوانی

نگارش :

علی ایزدی

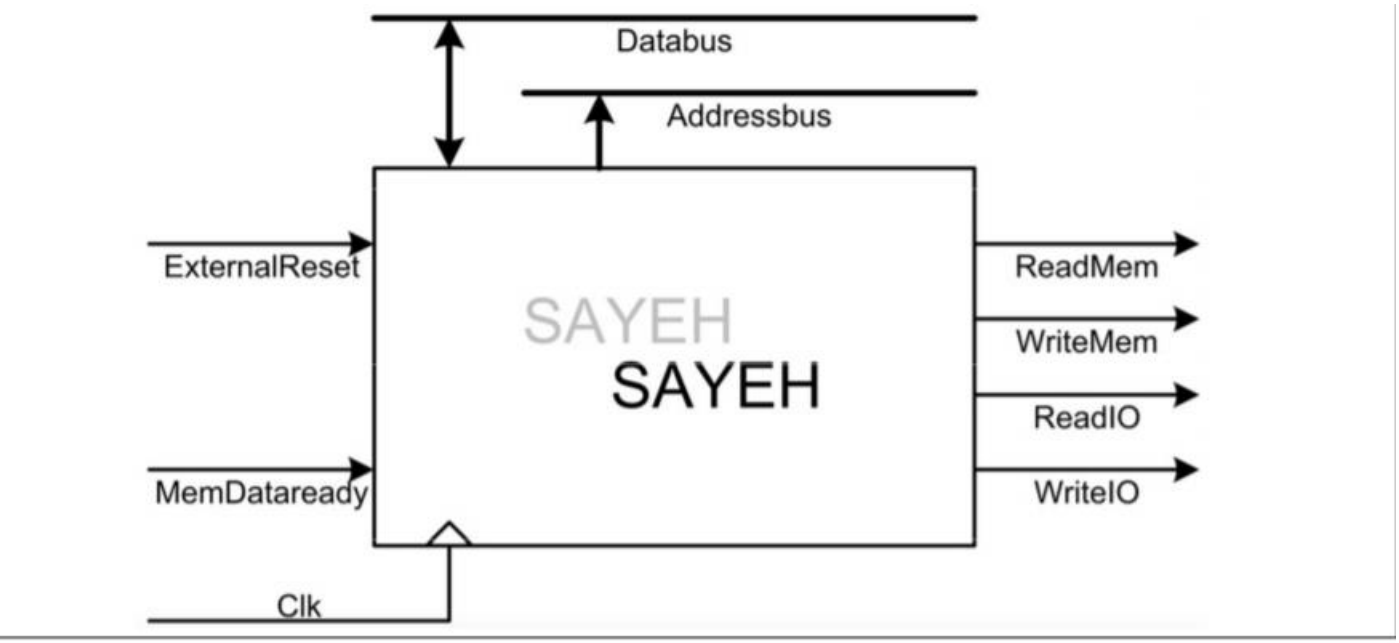
فروردین ماه ۱۳۹۶

چکیده

هدف از انجـام ایـن پـروژه طراحـی CPU سـایه شـامل component هـای مختلـف واحـد محاسبات، واحد دستورالعمل و سایر اجزای مورد نیاز این کامپیوتر و کنترل و ارتباط این اجـزا از طریق Controller و Data Path است.

این کامپیوتر دارای دستورات ۸ بیتی و ۱۶ بیتی است و وجود address bus ۱۶ بیتی امکان ترکیب دو دستور ۸ بیتی و خواندن این دو دستور از مموری ۱۶ بیتی در یک کـلاک را فـراهم میکند.

ماژول اصلی این CPU در تصویر زیر مشاهده میشود.



فهرست مطالب

-اجزای کامپیوتر(COMPONENTS)

-دستورالعمل ها (INSTRUCTIONS)

- DATAPATH

-CONTROLLER

-نتیجه گیری

اجزای کامپیوتر:

ALU:

بخش محاسباتی کامپیوتر که در آن عملیات زیر انجام میشود:
And, or, shift left, comparison, addition, subtraction

هم چنین بخش های امتیازی پیاده سازی شده به شرح زیر است:
Multiplication, Division, shift right, two's complement, XOR

واحد محاسباتی کامپیوتر اصلی (بدون بخش های امتیازی) شامل ۱۰ ورودی است که کنترلر وظیفه انتخاب کردن هر کدام برای مشخص کردن عملیات مربوطه را دارد.
این واحد هم چنین دو خروجی برای مشخص کردن flag های کامپیوتر دارد که وقتی خروجی ALU صفر است Zflag را یک و وقتی خروجی دارای carry out است Cflag را یک میکند. این دو flag در واحدی به نام **status register** که در زیر توضیح داده میشود نگه داری میشود.

STATUS REGSITER:

این بخش برای نگه داری Cout و Zout استفاده میشود. هم چنین این بخش قابلیت این را دارد که به صورت دستی flag ها set و clear شوند.

ADDRESSING UNIT:

این بخش شامل دو جز program counter و address logic است که pc رجیستری برای نگه داری آدرس خواندن از مموری در هر دوره زمانی اجرای یک دستور است و al مداری ترکیبی برای تغییر pc در عبارات شرطی با استفاده از دستورهای branch conditional و branch unconditional این CPU است.

INSTRUCITON REGISTER:

رجیستری است که برای نگهداری دستور خوانده شده از مموری استفاده میشود.

REGISTER FILE:

این مجموعه شامل ۶۴ رجیستر است که در هر لحظه میتوان به چهار رجیستر پشت سر هم آن مقدار داد. شماره ابتدای این ۴ رجیستر با ورودی ۶ بیتی window pointer مشخص میشود و شماره ۴ رجیستر نیز در خود دستور معین میگردد.
رجیستر فایل دارای دو ورودی RFlwrite و RFHwrite است که فعال کردن این دو تا به ترتیب باعث مقدار دادن ورودی به ۸ بیت کم ارزش و پرارزش رجیسترهای از قبل مشخص شده میشود.
رجیستر فایل دارای دو خروجی Rs و Rd است که برای انجام محاسبات رو رجیستر مقصد و مبدا انجام میگیرد.

WINDOW POINTER:

این رجیستر برای مشخص کردن شماره ابتدایی چهار رجیستر مبدا و مقصدی که پشت سر هم قرار گرفته اند استفاده میشود.
مقدار دهی آن نیز با دستور(awp (add window pointer انجام میگردد.

MEMORY:

ممووری این کامپیوتر دارای اندازه ۱۰۲۴ است که برای خواندن دستورها و ذخیره داده ها از آن استفاده میشود.

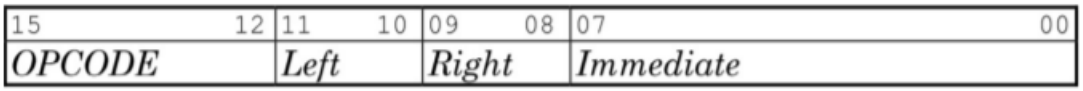
نکات :

-همه رجیسترها حساس به لبه کلاک هستند تا در لبه کلاک مقدار خود را تغییر دهند اما دو واحد alu و address logic به صورت آسنکرون بوده تا در لحظه مقدار خود را روی ترتیب به دیتاباس و آدرس باس بریزند.

-همه این اجزا باید با ماژولی به نام Data Path به یکدیگر سیم کشی شوند

-ورودی های همه ی این اجزا توسط ماژولی به نام controller کنترل میشوند تا دستورات مورد نیاز را بتوان طبق آن ها انجام داد.

دستورالعمل ها :



هر دستور العمل فرمی مطابق شکل بالا دارد.

قسمت left و right به ترتیب برای مشخص کردن یکی از چهار رجیستر ، رجیسترفایل است که به ترتیب برای رجیستر مقصد و مبدا استفاده میشوند.

قسمت opcode برای تعیین کردن نوع دستور استفاده میشود. هم چنین دستورهایی که نیاز به رجیستر مبدا یا مقصد را ندارند از قسمت left و right برای مشخص کردن دستورهای بیشتر استفاده میشود.

هر دستور ۱۶ تایی میتواند ترکیب دو دستور ۸ تایی باشد که shadow نامیده میشود. یعنی در هر fetch از مموری ابتدا دستور اول و سپس دستور دوم اجرا میشود.

قمست دوم که immediate نامیده میشود برای دستورهایی که شامل مقادیری برای انتقال به رجیسترها هستند استفاده میشود مانند: move immediate low

به طور کلی دستورات این کامپیوتر شامل دستورهای زیر میباشند:

-دستورهای محاسباتی توسط ALU
- دستورهایی برای set و clear رجیسترهای status
-دستورات no operation و hult
-دستورات انتقال به رجیسترفایل و مموری
- دستورات خواندن از ورودی و نوشتن در خروجی
- دستورات شرطی

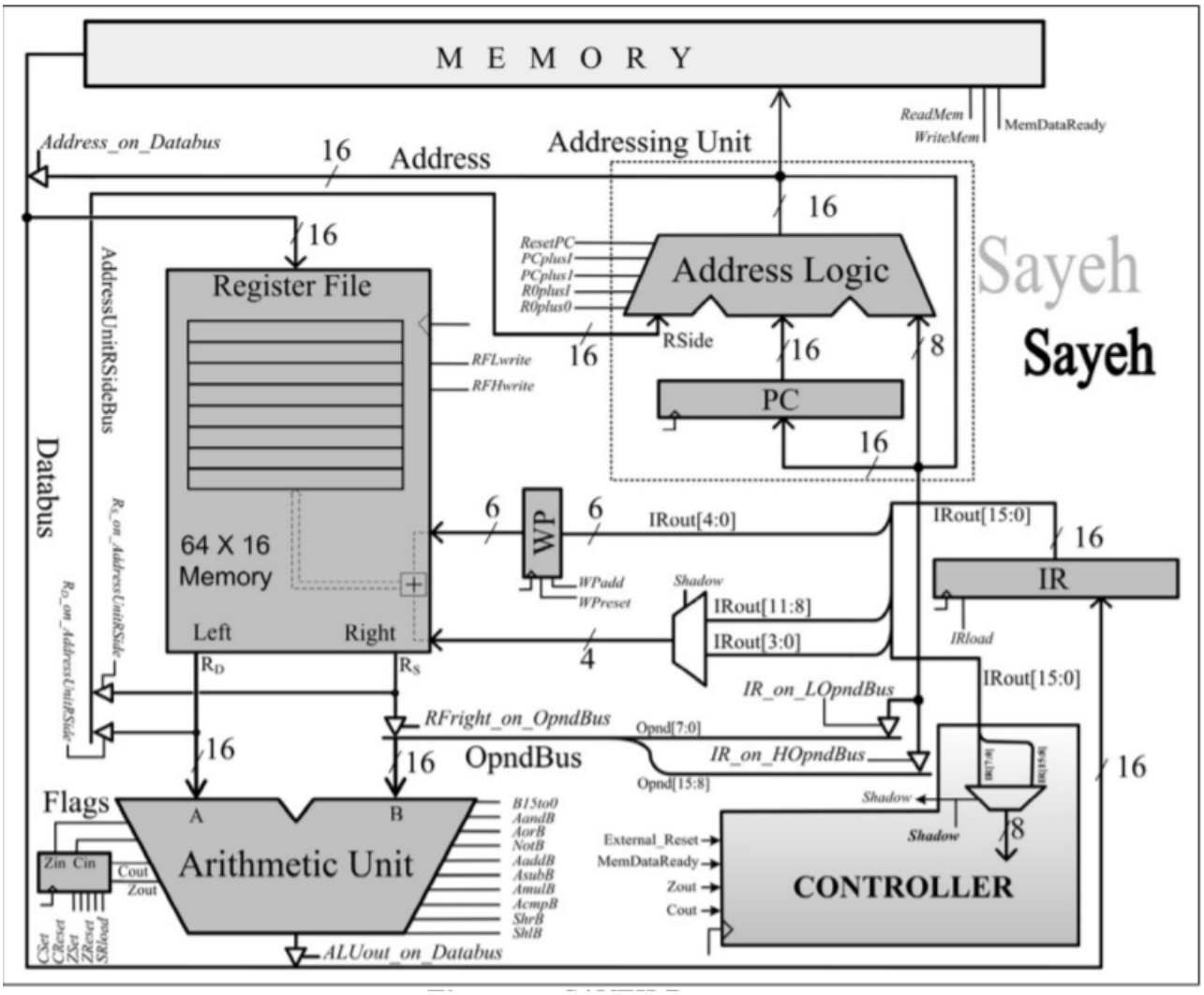
لیست همه دستورها به شرح زیر است :

Instruction Mnemonic and Definition		Bits 15:0	RTL notation: *comments or condition*
nop	No operation	0000-00-00	No operation
hlt	Halt	0000-00-01	Halt, fetching stops
szf	Set zero flag	0000-00-10	Z <= '1'
czf	Clr zero flag	0000-00-11	Z <= '0'
scf	Set carry flag	0000-01-00	C <= '1'
ccf	Clr carry flag	0000-01-01	C <= '0'
cwp	Clr Window pointer	0000-01-10	WP <= "000"
mvr	Move Register	0001-D-S	R_D <= R_S
lda	Load Addressed	0010-D-S	R_D <= (R_S)
sta	Store Addressed	0011-D-S	(R_D) <= R_S
inp	Input from port	0100-D-S	In from R_S write to R_D
oup	Output to port	0101-D-S	Out to port R_D from R_S
and	AND Registers	0110-D-S	R_D <= R_D & R_S
orr	OR Registers	0111-D-S	R_D <= R_D | R_S
not	NOT Register	1000-D-S	R_D <= ~R_S
shl	Shift Left	1001-D-S	R_D <= sla R_S
shr	Shift Right	1010-D-S	R_D <= sra R_S
add	Add Registers	1011-D-S	R_D <= R_D + R_S + C
sub	Subtract Registers	1100-D-S	R_D <= R_D - R_S – C
mul	Multiply Registers	1101-D-S	R_D <= R_D * R_S :*8-bit multiplication*
cmp	Compare	1110-D-S	R_D, R_S *(if equal:Z=1; if R_D<R_S: C=1)*
mil	Move Immd Low	1111-D-00-I	R_{DL} <= {8'bZ, I}
mih	Move Immd High	1111-D-01-I	R_{DH} <= {I, 8'bZ }
spc	Save PC	1111-D-10-I	R_D <= PC + I
jpa	Jump Addressed	1111-D-11-I	PC <= R_D + I
jpr	Jump Relative	0000-01-11-I	PC <= PC + I
brz	Branch if Zero	0000-10-00-I	PC <= PC + I :*if Z is 1*
brc	Branch if Carry	0000-10-01-I	PC <= PC + I :*if C is 1*
awp	Add Win pntr	0000-10-10-I	WP <= WP + I

Data path:

این ماژول شامل سیم کشی همه کامپوننت های تعریف شده است.
در این ماژول از بافرهایی استفاده شده است تا مشخص کنند در هر لحظه خروجی کدام یک از اجزا روی data bus یا address bus قرار گیرند.

مطابق شکل زیر اجزا به یکدیگر وصل شده اند :



در زیر کد vhdl مربوط به وصل کردن اجزای مختلف در data path آمده است :

```
AddressUnitRSideBus <= Right when Rs_on_AddressUnitRSide='1' else
                       Left  when Rd_on_AddressUnitRSide='1' else
                       "ZZZZZZZZZZZZZZZZ";

Addressbus <= Address;

Databus <= Address when Address_on_Databus ='1' else
           ALUout  when ALU_on_Databus='1' else
           "ZZZZZZZZZZZZZZZZ";



OpndBus(7 downto 0) <= IRout(7 downto 0) when IR_on_LOpndBus='1' else
                       "ZZZZZZZZ" ;

OpndBus(15 downto 8) <= IRout (7 downto 0) when IR_on_HOpndBus ='1' else
                        "ZZZZZZZZ" ;

OpndBus <=  Right when RFright_on_OpndBus ='1' else
            "ZZZZZZZZZZZZZZZZ";
Zout <= SRZout;
Cout <= SRCout;

Instruction <= IRout(15 downto 0);

ShadowEn <= '0' when IRout (7 downto 0 ) = "00001111" else '1' ;

Laddr <= IRout(11 downto 10) when Shadow='0' else
         IRout(3 downto 2) ;
Raddr <= IRout(9 downto 8) when Shadow='0' else
         IRout(1 downto 0) ;
```

:CONTROLLER

این بخش برای تصمیم گیری برای انجام دستورات است.
کنترلر با استفاده از ماشین حالت متناهی FSM تصمیم میگیرد که برای انجام هر دستور مشخص کدام یک از سیگنال های ورودی DATA PATH فعال شوند تا عملیات مربوطه انجام گردد.
استیت ماشین استفاده شده در کنترلر شامل ۱۱ استیت به شرح زیر اند:

-RESET: با استفاده از سیگنال ExternalReset وارد این استیت میشویم

-HALT: در این استیت کامپیوتر متوقف میشود تا زمانی که ریست شود.

-FETCH: در این مرحله سیگنال خواندن از حافظه فعال میشود تا مقدار حافظه روی دیتاباس قرارگیرد

-MEMREAD: در این مرحله مقدار از حافظه خوانده شده و اکنون دیتاباس مقدار حافظه را روی خود دارد.

-EXE1: این مرحله برای انجام ۸ بیت اول دستور انجام میشود

-EXE2: این مرحله برای انجام ۸ بیت دوم دستور و برای هندل کردن دستورات Shadow انجام میگیرد.

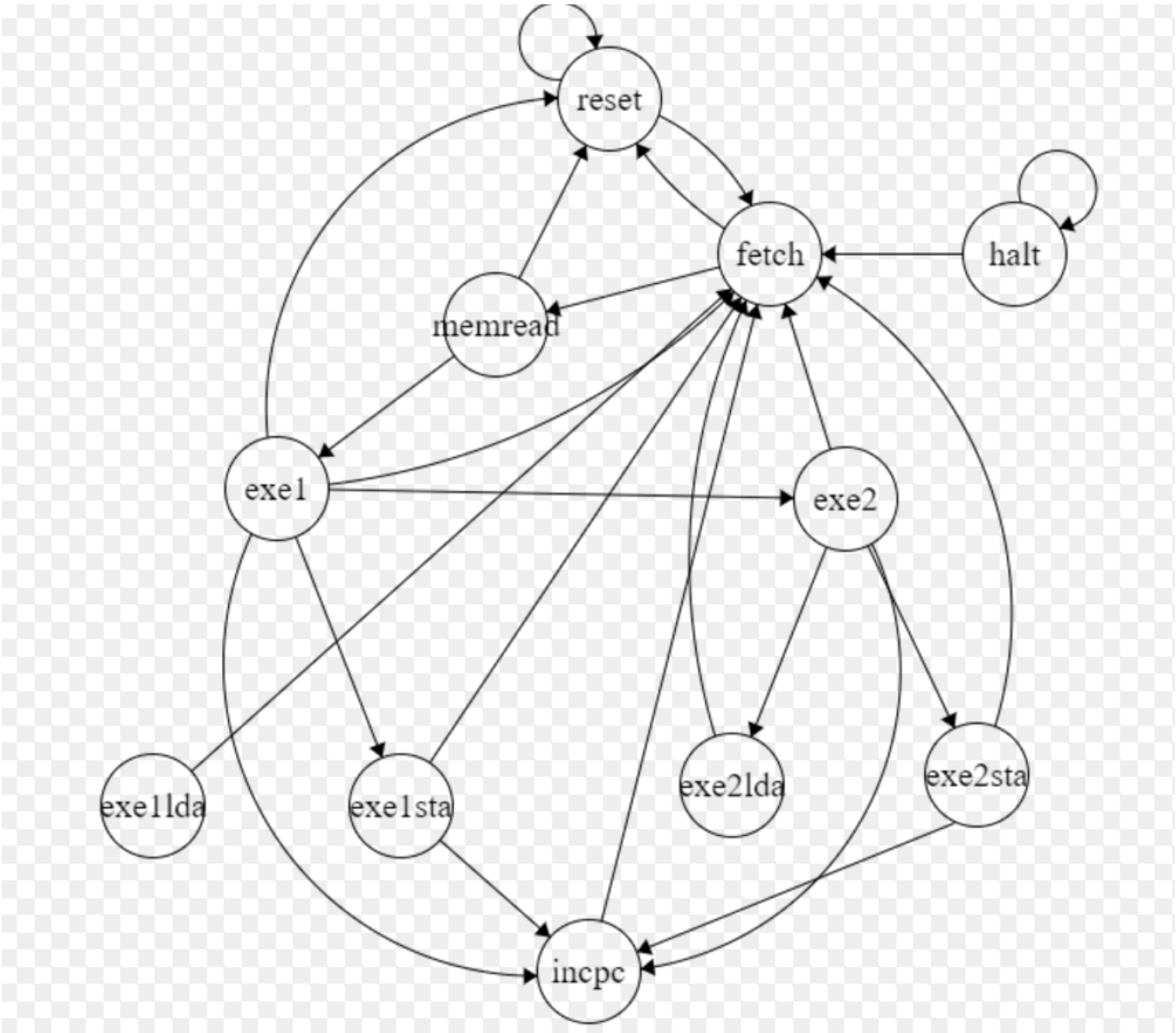
-EXE1LDA: این مرحله برای دستور load addressed انجام میگرد تا بتوان کلاک اضافه مورد نیاز برای خواندن از حافظه و ریختن مقدار حافظه در رجیسترفایل را هندل کرد.

-EXE2LDA: این مرحله مطابق مرحله قبل ولی برای انجام دستور LOAD زمانی که در ۸ بیت دوم قرار میگیرد استفاده میشود.

-INCPC: این استیت نیز برای انجام دستوراتیی که نیاز به یک کلاک اضافه تر برای افزایش program counter را دارند استفاده میشود

-EXE1STA , EXE2STA: مشابه دو دستور برای LOAD این دو مرحله انجام دستور store addressed استفاده میشوند.

FSM طراحی شده مطابق شکل زیر است :



تست :

در پایان تست با مقدار دهی مموری با استفاده از دستورات زیر انجام گرفت :

```
-- cwp
buffermem(0) := "00000000_00000110"

-- mil r0, 01011101
buffermem(1) := "11110000_01011101"

-- mih r0, 00000101
buffermem(2) := "11110001_00000101"

-- mil r1, 00000001
buffermem(3) := "11110100_00000001"

-- mih r1, 00000000
buffermem(4) := "11110101_00000000"

-- add r1, r0
buffermem(5) := "00000000_10110100"
```

نتیجه گیری:

پردازنده سایه که توسط دکتر نوابی طراحی شده است با استفاده از زبان طراحی سخت افزار VHDL به طور کامل در تاریخ ۱ اردیبهشت تست وانجام آن به اتمام رسید.