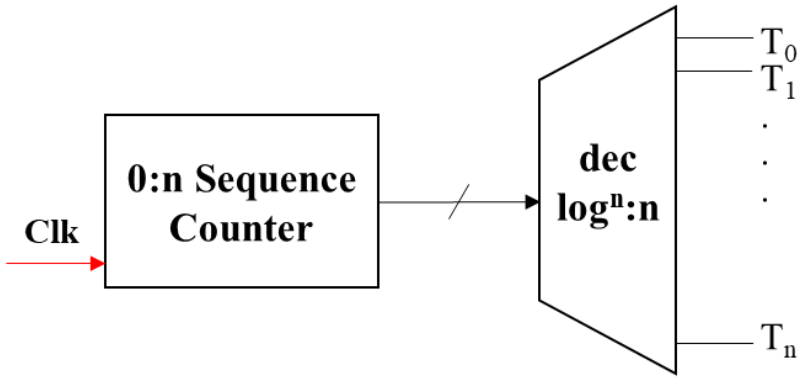
بنام خداوند جان و خرد

معماری کامپیوتر

جلسه بیست‌ویکم: واحد کنترل (Control Unit)-۴

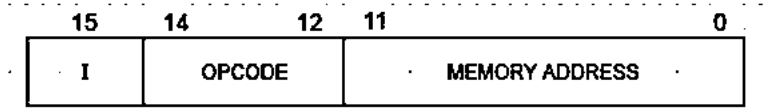
طراحی توالی‌گر در واحد کنترل

- به‌منظور اعمال الگوریتم نیومن در اجرای دستورات
  - نیاز به اجرای ترتیبی در سخت‌افزار داریم
    - نیاز به طراحی واحد سخت‌افزاری با هدف ایجاد توالی: توالی‌گر (sequencer)
    - در هر زمان، طبق روال مشخص یکی از مراحل فعال و سایرین غیرفعال باشند

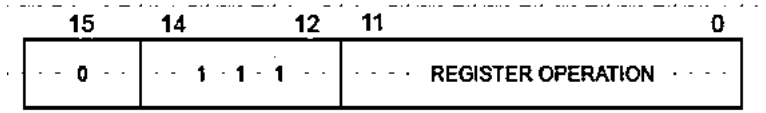


مجموعه دستورالعمل‌ها در کامپیوتر پایه

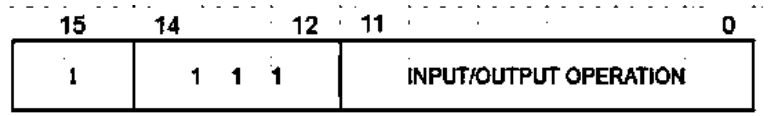
- قالب دستورالعمل



- تعداد دستورات حافظه‌ای
  - هفت دستور با آدرس‌دهی مستقیم و هفت دستور با آدرس‌دهی غیرمستقیم



- تعداد دستورات ثباتی
  - دوازده دستور به تعداد بیت‌های آدرس (هربار یکی از بیت‌های آدرس یک باشد)
  - چون تعداد ثبات‌ها محدود است، از بخش آدرس به‌عنوان کدعملیاتی استفاده می‌کنیم



- دستورات ورودی خروجی
  - دوازده بیت برای تعیین پورت

دستورات قابل اجرا در کامپیوتر پایه

	Symbol	Hex code	Description
Memory	AND	0 or 8	AND M to AC
	ADD	1 or 9	Add M to AC, carry to E
	LDA	2 or A	Load AC from M
	STA	3 or B	Store AC in M
	BUN	4 or C	Branch unconditionally to m
	BSA	5 or D	Save return address in m and branch to m+1
	ISZ	6 or E	Increment M and skip if zero
Register	CLA	7800	Clear AC
	CLE	7400	Clear E
	CMA	7200	Complement AC
	CME	7100	Complement E
	CIR	7080	Circulate right E and AC
	CIL	7040	Circulate left E and AC
	INC	7020	Increment AC, carry to E
	SPA	7010	Skip if AC is positive
	SNA	7008	Skip if AC is negative
	SZA	7004	Skip if AC is zero
	SZE	7002	Skip if E is zero
	HLT	7001	Halt computer
I/O	INP	F800	Input information and clear flag
	OUT	F400	Output information and clear flag
	SKI	F200	Skip if input flag is on
	SKO	F100	Skip if output flag is on
	ION	F080	Turn interrupt on
	IOF	F040	Turn interrupt off

دستورات قابل اجرا در کامپیوتر پایه

- دستورات قابل اجرا به چهار دسته قابل تقسیم می‌باشند:
  - دستورات عملیاتی (Functional Instructions)
    - محاسبات، منطق، عملیات شیفت
    - ADD, AND, CMA, CME, INC, CIL, CIR
  - دستورات انتقال (Transfer Instructions)
    - انتقال بین حافظه اصلی و ثبات‌ها
    - LDA, STA
  - دستورات کنترلی (Control Instructions)
    - توالی اجرای برنامه و کنترل مانند jumpها
  - دستورات ورودی–خروجی (Input-Output Instructions)
    - ارتباط با دستگاه‌های ورودی/خروجی

طراحی واحد کنترل

- می‌دانیم در کدام مرحله اجرایی هستیم (sequencer) پس باید عملیات آن مرحله را بسازیم
  - چه ثبات‌هایی فعال می‌شوند (مدیریت باس با MUX)
  - نحوه ارتباط ثبات‌ها با یکدیگر، حافظه و ALU
  - شیوه فعال شدن ورودی‌های load و ... ثبات‌ها

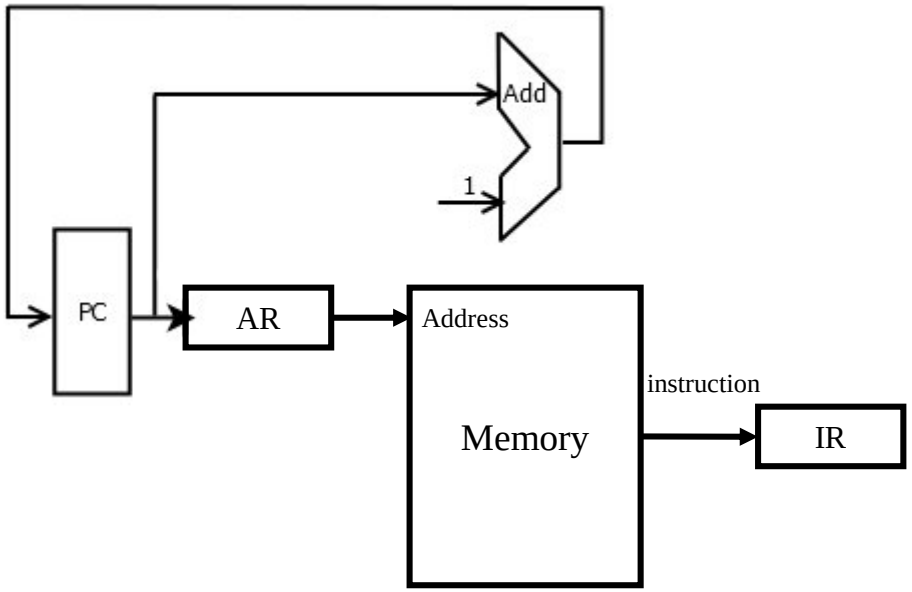
طراحی اجرای الگوریتم فون نیومن

- پیاده‌سازی چرخه اجرای دستورالعمل‌ها (روش استفاده از Microinstructions)
  - Microinstruction (ریزدستورالعمل):
    - دستوری که جریان داده را کنترل می‌کند
    - هر دستور برای اجرا، به دنباله‌ای از ریزدستورات تقسیم می‌شود
    - فرمت ریزدستورالعمل: **شرط: R ← operation**
      - درصورت برآورده شدن شرط، عملیات را انجام بده و حاصل را در ثبات R بریز
      - در این روش، شرط را براساس مراحل الگوریتم نیومن تنظیم می‌کنیم

طراحی اجرای الگوریتم فون نیومن

۱- مرحله Instruction Fetch

- خواندن PC و نوشتن آن در AR
- T0: AR←PC
- نوشتن دستورالعمل جاری در IR
- T1: IR ← M[AR], PC++



طراحی اجرای الگوریتم فون نیومن

۲- مرحله Decode

- کدگشایی بیت‌های ۱۲ الی ۱۴ دستورالعمل
- تعیین نوع آدرس‌دهی با بیت ۱۵ دستورالعمل
- قرار دادن بیت‌های ۰ الی ۱۱ در AR جهت استخراج داده

T2: D0:D7 ← Decode IR(12:14),AR ← IR(0:11), I ← IR(15)

طراحی اجرای الگوریتم فون نیومن

۳- مرحله خواندن عملوندها

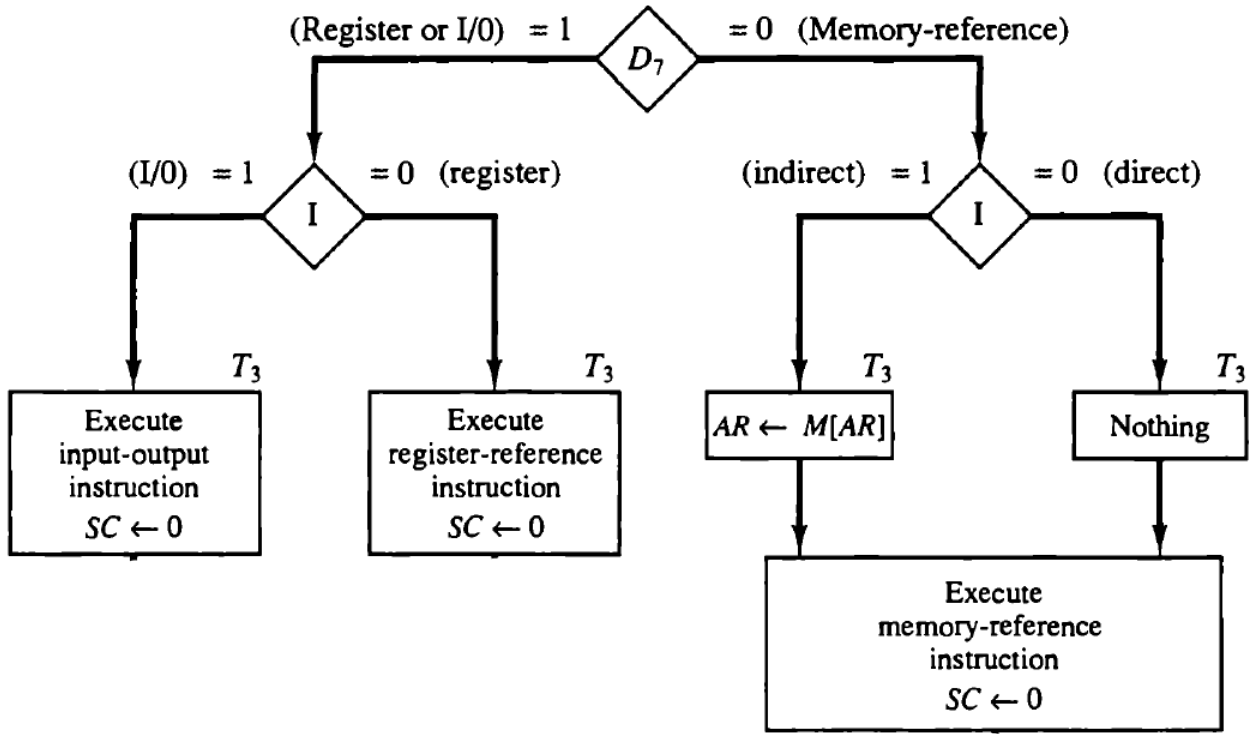
- مرحله خواندن عملوند مخصوص دستورات حافظه‌ای است
- دستورات ثباتی و ورودی و خروجی این مرحله را ندارند

۴- مرحله Execute

- براساس بیت D7 مرحله دوم، نوع دستور مشخص شده و اجرا می‌گردد
  - برای دستورات مختلف، ریزدستورالعمل‌های متفاوت داریم
- توالی‌گر پس این مرحله ریست می‌شود

طراحی اجرای الگوریتم فون نیومن

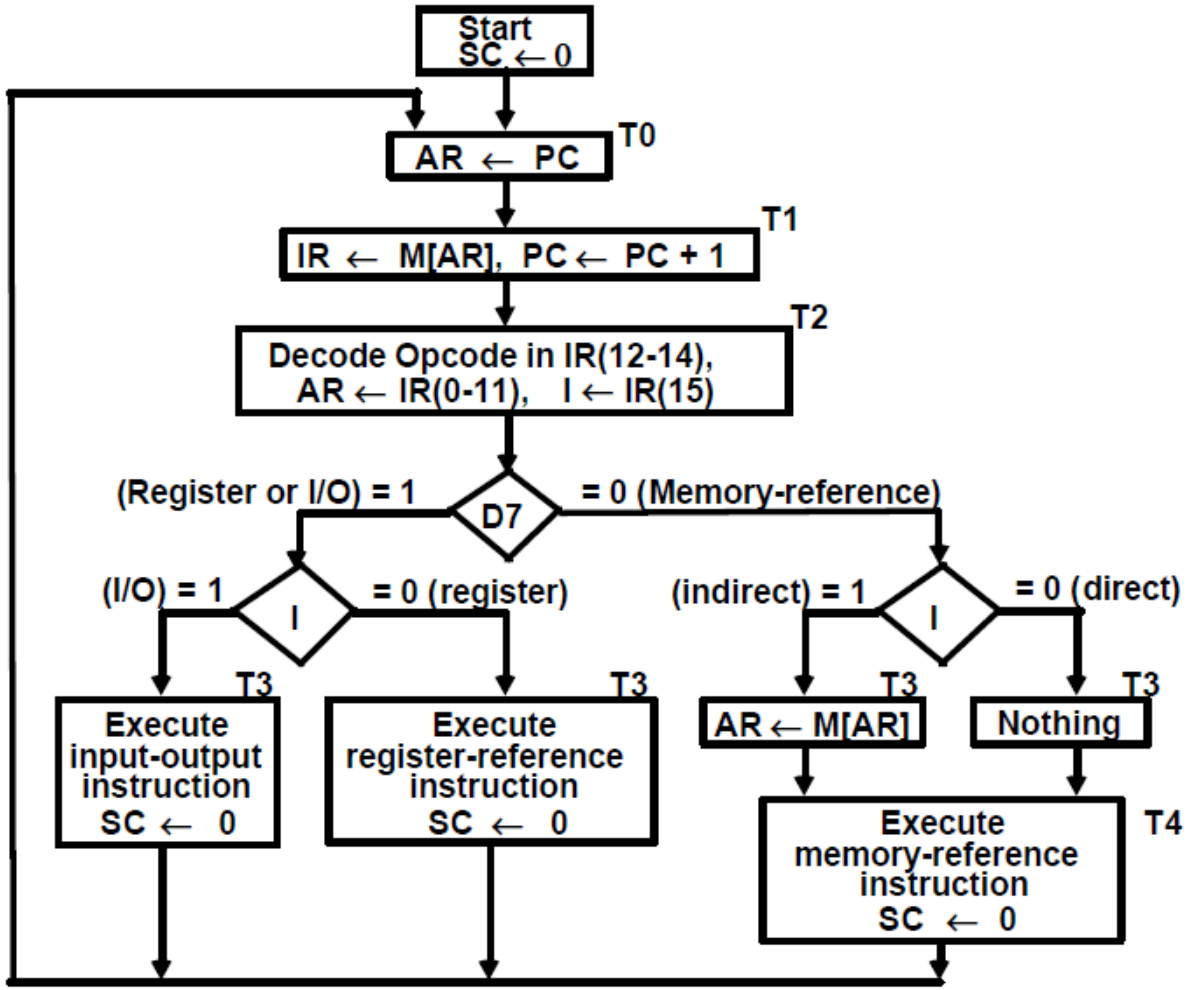
- در مراحل ۳ و ۴ داریم:



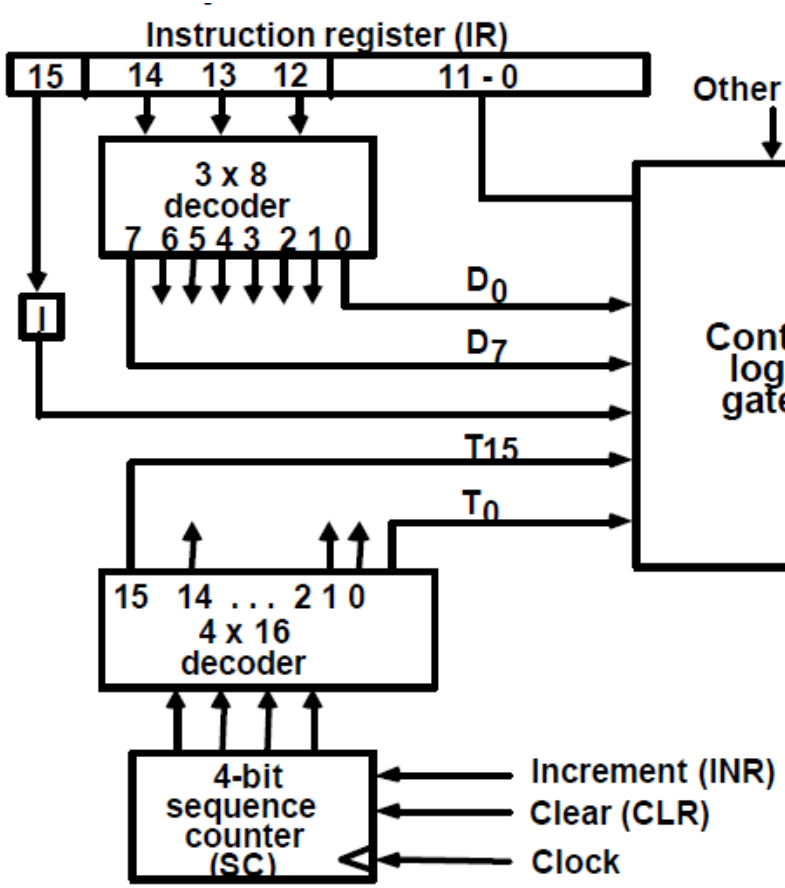
طراحی اجرای الگوریتم فون نیومن

- در دستورات ثباتی و I/O، مراحل ۳ و ۴ در یک کلاک اجرا می‌شوند
  - نیاز به واکشی عملوندها نمی‌باشد
- در دستورات حافظه‌ای، مراحل ۳ و ۴ در بیش از یک کلاک اجرا می‌شوند
- چرخه ۴ مرحله‌ای شرح داده شده ← instruction cycle
  - Machine cycle: کلاک پردازنده

فلوچارت مراحل چرخه دستورالعمل



واحد کنترل کامپیوتر پایه



- تنظیم ورودی‌های LD,CLR,IN
- مدیریت باس (MUX)

دستورات قابل اجرا در کامپیوتر پایه

- در دستورات ثباتی باید شرط $D_7I'T_3$ برقرار باشد
- در دستورات بیان شده علاوه برشرط بالا باید یکی از بیت‌های بخش عملوند هم یک بود
- مثلا در دستور CLA داریم: $D_7I'T_3B[11]$:AC←0 شرط

	r:	SC ← 0
CLA	rB_{11}:	AC ← 0
CLE	rB_{10}:	E ← 0
CMA	rB_9:	AC ← AC'
CME	rB_8:	E ← E'
CIR	rB_7:	AC ← shr AC, AC(15) ← E, E ← AC(0)
CIL	rB_6:	AC ← shl AC, AC(0) ← E, E ← AC(15)
INC	rB_5:	AC ← AC + 1
SPA	rB_4:	if (AC(15) = 0) then (PC ← PC+1)
SNA	rB_3:	if (AC(15) = 1) then (PC ← PC+1)
SZA	rB_2:	if (AC = 0) then (PC ← PC+1)
SZE	rB_1:	if (E = 0) then (PC ← PC+1)
HLT	rB_0:	S ← 0 (S is a start-stop flip-flop)

دستورات قابل اجرا در کامپیوتر پایه

- در دستورات حافظه‌ای، یکی از خروجی‌های D_0 تا D_6 یک است
- اجرای برخی دستورات حافظه‌ای بیش از یک کلاک نیاز دارد
  - D_0T_4: DR←M[AR] (چرا این مرحله لازم است؟)
  - D_0T_5: AC ← AC & DR,Sc←0

Symbol	Operation Decoder	Symbolic Description
AND	D_0	AC ← AC ∧ M[AR]
ADD	D_1	AC ← AC + M[AR], E ← C_{out}
LDA	D_2	AC ← M[AR]
STA	D_3	M[AR] ← AC
BUN	D_4	PC ← AR
BSA	D_5	M[AR] ← PC, PC ← AR + 1
ISZ	D_6	M[AR] ← M[AR] + 1, if M[AR] + 1 = 0 then PC ← PC+1

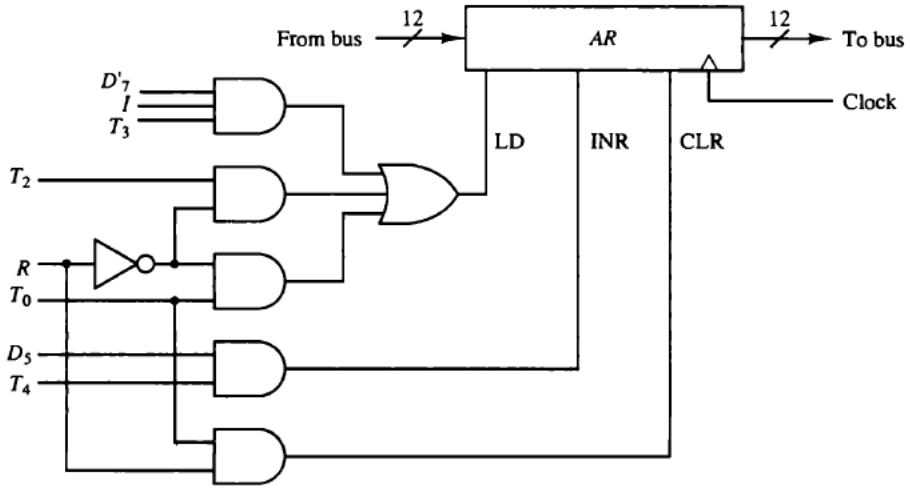
دستورات قابل اجرا در کامپیوتر پایه

Symbol	Operation Decoder	Symbolic Description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1$, if $M[AR] + 1 = 0$ then $PC \leftarrow PC+1$

- بارگذاری (load) ثبات AC از طریق DR انجام می‌شود
  - D_2T_4: DR←M[AR]
  - D_2T_5: AC ← DR, Sc←0
- دستورالعمل ISZ
  - D_6T_4: DR←M[AR]
  - D_6T_5: DR ← DR+1
  - D_6T_6: M[AR]=DR, if (DR=0) then PC←PC+1, Sc←0

واحد کنترل کامپیوتر پایه

- در انتها براساس ریزدستورات، پایه‌های متصل به باس را برنامه‌ریزی می‌کنیم
  - مثلا همه زمان‌هایی که AR مقداردهی می‌شود را باهم OR می‌کنیم
  - نتیجه این OR را به پایه load ثبات AR روی باس می‌دهیم



- $LD[AR] \leftarrow R'T_0+R'T_2+D_7'IT_3$

$AR \leftarrow PC,\ AR \leftarrow IR(0\text{-}11),AR \leftarrow M[AR]$

- $CLR[AR] \leftarrow RT_0$
- $INR[AR] \leftarrow D_5T_4$