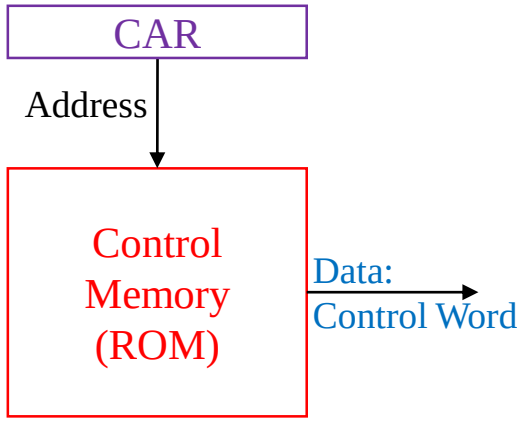
بنام خداوند جان و خرد

معماری کامپیوتر

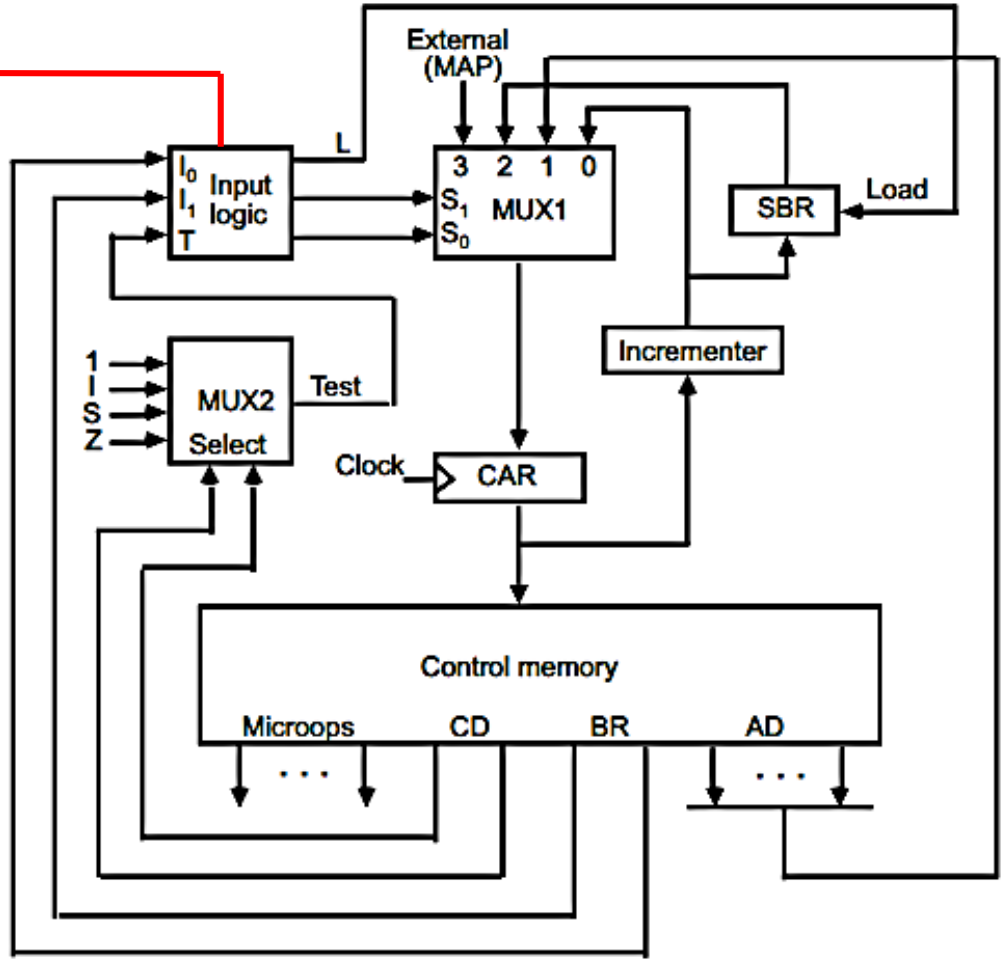
جلسه بیست‌وپنجم: کنترل ریزبرنامه‌ریزی شده۳

ساختار و روال عملکرد حافظه کنترلی

- از جنس ROM است و یک بخش آدرس و یک بخش داده دارد
- روال عملکرد:
  - آدرس ریزدستورالعمل جاری به حافظه کنترلی داده می‌شود (CAR)
  - به مکانی که آدرس مشخص شده رفته و داده آن را استخراج می‌شود (شیوه ذخیره داده مهم است)
  - قسمتی از داده که مربوط به اطلاعات کنترلی است در اختیار مسیر داده قرار می‌گیرد
  - آدرس بعدی تولید می‌شود (address sequencer)



واحد تولید آدرس بعدی



Input Logic

I_0I_1T	Meaning	Source of Address	S $_1S_0$	L
000	In-Line	CAR+1	00	0
001	JMP	CS(AD)	10	0
010	In-Line	CAR+1	00	0
011	CALL	CS(AD) and SBR <- CAR+1	10	1
10x	RET	SBR	01	0
11x	MAP	DR(11-14)	11	0

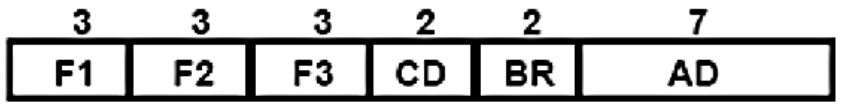
$S_0 = I_0$

$S_1 = I_0I_1 + I_0'T$

$L = I_0'I_1T$

ذخیره داده (microinstruction) در حافظه کنترلی

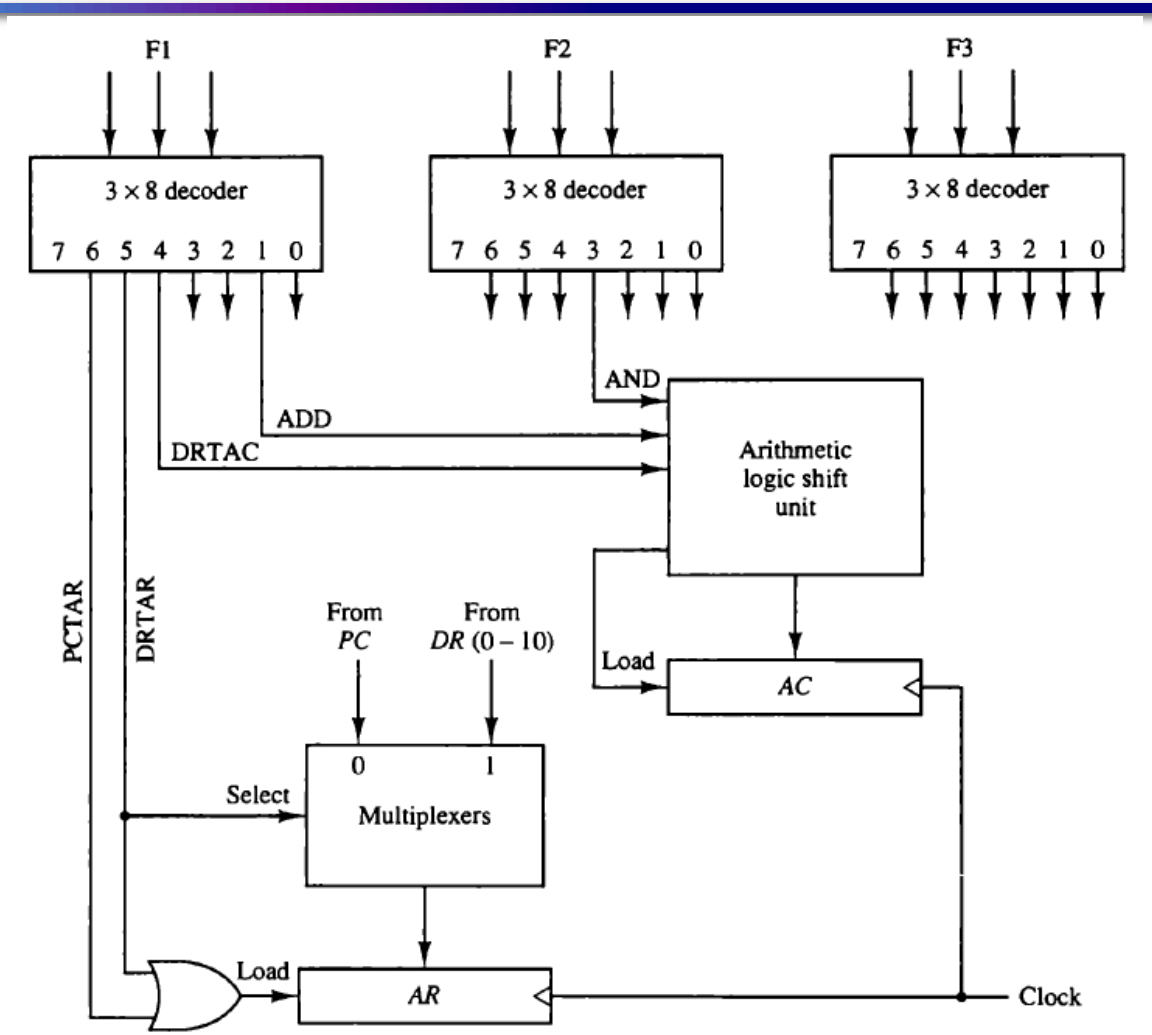
- در هر خط حافظه ۲۰ بیت ذخیره می‌شد
  - فیلد AD: آدرس بعدی را تولید می‌کند که ۷ بیتی است
  - فیلد BR: نوع پرش را مشخص کند
  - فیلد CD: شرط پرش را مشخص می‌کند
  - فیلدهای F: به مسیر داده می‌رود و ریزعملیات را مشخص می‌کند
    - توسط دیکدر مسیر داده را مشخص می‌کنند



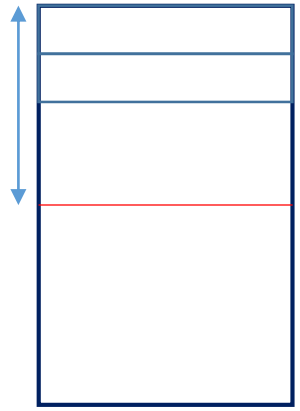
F1, F2, F3: Microoperation fields
CD: Condition for branching
BR: Branch field
AD: Address field

ساخت کلمه کنترلی از داده‌های حافظه کنترلی

F1	Microoperation	Symbol
000	None	NOP
001	AC ← AC + DR	ADD
010	AC ← 0	CLRAC
011	AC ← AC + 1	INCAC
100	AC ← DR	DRTAC
101	AR ← DR(0-10)	DRTAR
110	AR ← PC	PCTAR
111	M[AR] ← DR	WRITE

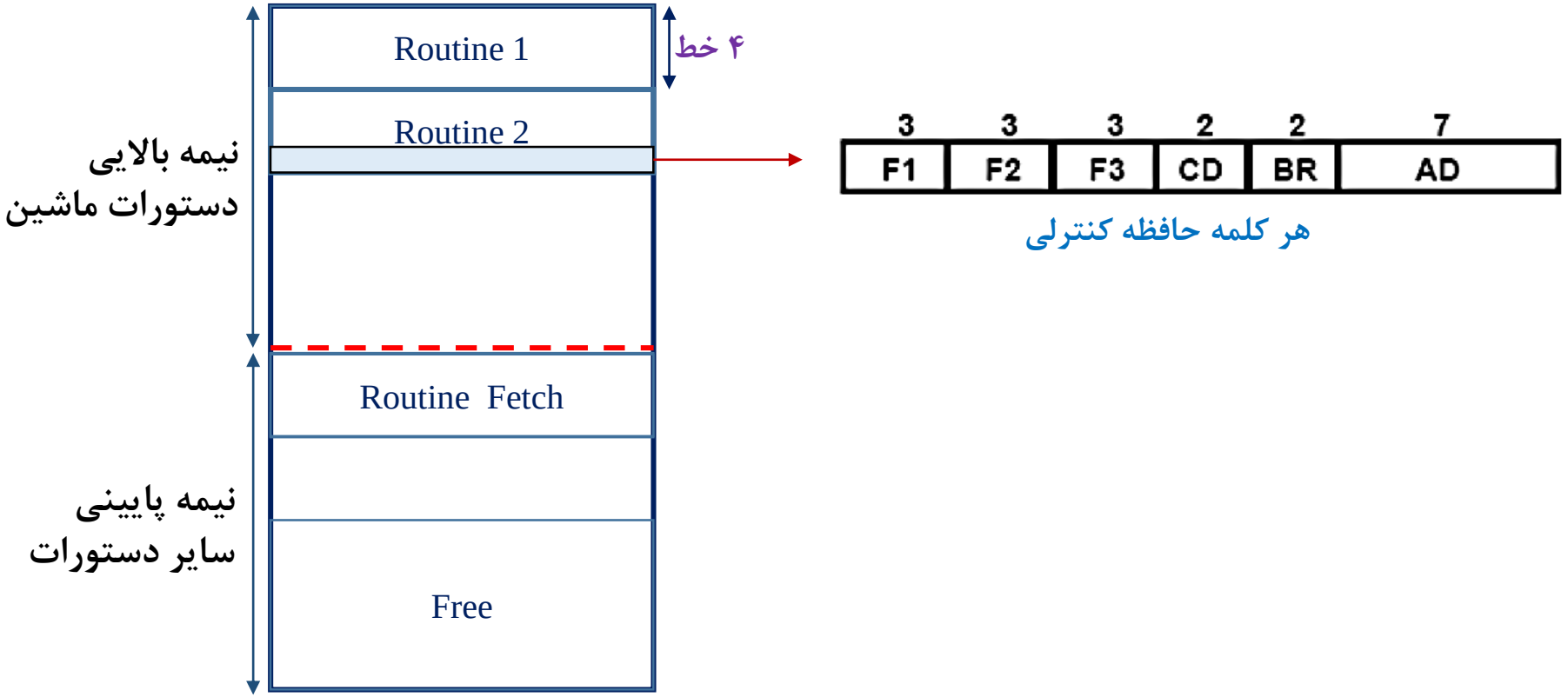


شیوه ذخیره داده در حافظه کنترلی



- گفتیم که حافظه کنترلی متشکل از ۱۲۸ کلمه (خط) است
  - در طراحی، ۶۴ بیت اول برای دستورات ماشین (اجرایی) درنظر گرفته شده است
    - گفتیم برای هریک از این دستورات، یک روتین چهار خطی داریم
  - ۶۴ بیت دوم آزاد برای دستورات عملیات (مانند fetch) یا دستورات طولانی درنظر گرفته شده است
- شیوه نوشتار ریزدستورالعمل‌ها، براساس زبان اسمبلی است
  - این دستورات سپس توسط اسمبلر به کد باینری تبدیل و اجرا می‌شود
    - کدهای باینری مطابق جداولی است که برای بخش‌های داده دیدیم (اسلایدهای ۶، ۸ و ۹ جلسه قبل)

شیوه ذخیره داده در حافظه کنترلی



شیوه ذخیره داده در حافظه کنترلی

- برای نوشتن ریزدستورات، ۵ بخش را مشخص می‌کنیم
  - **Label:** مشخص می‌کند روتین فعلی برای چه عملیاتی است و پس از نام آن دونقطه داریم
  - **Micro-ops:** شامل سه تابع است که نشانگر ریزعملگرها می‌باشند (فیلدهای F) و با کاما جدا می‌شوند
  - **CD:** شرط پرش را مشخص می‌کند و چهار حالت داشتیم: U,I,S,Z
  - **BR:** نوع پرش را مشخص می‌کند و چهار حالت داشتیم: JMP, Call, Ret, Map
  - **AD:** آدرس پرش را مشخص می‌کند
  - اشاره به یک label، Next (ریزدستور العمل بعدی)، یا هیچ برای زمانی که BR دستور بعدی را نشان می‌دهد

شیوه ذخیره داده در حافظه کنترلی

- مطابق چرخه دستورالعمل (instruction cycle)
- ابتدا به fetch می‌رویم که در این طراحی، در ابتدای بخش دوم حافظه قرار داده شده است
- عملیات fetch

$$AR \leftarrow PC$$
$$DR \leftarrow M[AR], \quad PC \leftarrow PC + 1$$
$$AR \leftarrow DR(0\text{–}10), \quad CAR(2\text{–}5) \leftarrow DR(11\text{–}14), \quad CAR(0,1,6) \leftarrow 0$$

- شیوه ذخیره روتین fetch در حافظه کنترلی

```
        ORG 64
FETCH:  PCTAR           U    JMP    NEXT
        READ, INCPC     U    JMP    NEXT
        DRTAR           U    MAP
```

شیوه ذخیره داده در حافظه کنترلی-تعیین عملوند و اجرا

Symbol	OP-code	Description
ADD	0000	AC ← AC + M[EA]
BRANCH	0001	if (AC < 0) then (PC ← EA)
STORE	0010	M[EA] ← AC
EXCHANGE	0011	AC ← M[EA], M[EA] ← AC

Label	Microops	CD	BR	AD
	ORG 0			
ADD:	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ADD	U	JMP	FETCH
	ORG 4			
BRANCH:	NOP	S	JMP	OVER
	NOP	U	JMP	FETCH
OVER:	NOP	I	CALL	INDRCT
	ARTPC	U	JMP	FETCH
	ORG 8			
STORE:	NOP	I	CALL	INDRCT
	ACTDR	U	JMP	NEXT
	WRITE	U	JMP	FETCH
	ORG 12			
EXCHANGE:	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ACTDR, DRTAC	U	JMP	NEXT
	WRITE	U	JMP	FETCH
	ORG 64			
FETCH:	PCTAR	U	JMP	NEXT
	READ, INCPC	U	JMP	NEXT
	DRTAR	U	MAP	
INDRCT:	READ	U	JMP	NEXT
	DRTAR	U	RET	

Fetch Data indirectly

ابتدای چرخه دستورالعمل

شیوه ذخیره داده در حافظه کنترلی

Micro Routine	Address		Binary Microinstruction					
	Decimal	Binary	F1	F2	F3	CD	BR	AD
ADD	0	0000000	000	000	000	01	01	1000011
	1	0000001	000	100	000	00	00	0000010
	2	0000010	001	000	000	00	00	1000000
	3	0000011	000	000	000	00	00	1000000
BRANCH	4	0000100	000	000	000	10	00	0000110
	5	0000101	000	000	000	00	00	1000000
	6	0000110	000	000	000	01	01	1000011
	7	0000111	000	000	110	00	00	1000000
STORE	8	0001000	000	000	000	01	01	1000011
	9	0001001	000	101	000	00	00	0001010
	10	0001010	111	000	000	00	00	1000000
	11	0001011	000	000	000	00	00	1000000
EXCHANGE	12	0001100	000	000	000	01	01	1000011
	13	0001101	001	000	000	00	00	0001110
	14	0001110	100	101	000	00	00	0001111
	15	0001111	111	000	000	00	00	1000000
FETCH	64	1000000	110	000	000	00	00	1000001
	65	1000001	000	100	101	00	00	1000010
	66	1000010	101	000	000	00	11	0000000
INDRCT	67	1000011	000	100	000	00	00	1000100
	68	1000100	101	000	000	00	10	0000000

شیوه ذخیره داده در حافظه کنترلی-تعیین عملوند و اجرا

- مثال: ریزعملگرهای متناظر ریزدستورالعمل‌های زیر را در کامپیوتر نمونه مشخص کنید.

- AC ← AC + 1 , PC ← PC + 1
  - INCAC, NOP, INCPC → 011000101
- DR ← M[AR] , PC ← AR
  - NOP, READ, ARTPC → 000100110

F1	Microoperation	Symbol	F2	Microoperation	Symbol	F3	Microoperation	Symbol
000	None	NOP	000	None	NOP	000	None	NOP
001	AC ← AC + DR	ADD	001	AC ← AC - DR	SUB	001	AC ← AC ⊕ DR	XOR
010	AC ← 0	CLRAC	010	AC ← AC ∨ DR	OR	010	AC ← AC'	COM
011	AC ← AC + 1	INCAC	011	AC ← AC ∧ DR	AND	011	AC ← shl AC	SHL
100	AC ← DR	DRTAC	100	DR ← M[AR]	READ	100	AC ← shr AC	SHR
101	AR ← DR(0-10)	DRTAR	101	DR ← AC	ACTDR	101	PC ← PC + 1	INCPC
110	AR ← PC	PCTAR	110	DR ← DR + 1	INCDR	110	PC ← AR	ARTPC
111	M[AR] ← DR	WRITE	111	DR(0-10) ← PC	PCTDR	111	Reserved	

شیوه ذخیره داده در حافظه کنترلی-تعیین عملوند و اجرا

- مثال: روتین نمونه زیر چه عملیاتی انجام می‌دهد؟

```
ORG  40
NOP       S       JMP       FETCH
NOP       Z       JMP       FETCH
NOP       I       CALL      INDRCT
ARTPC     U       JMP       FETCH
```

حل: IF (AC>0) then PC←AR