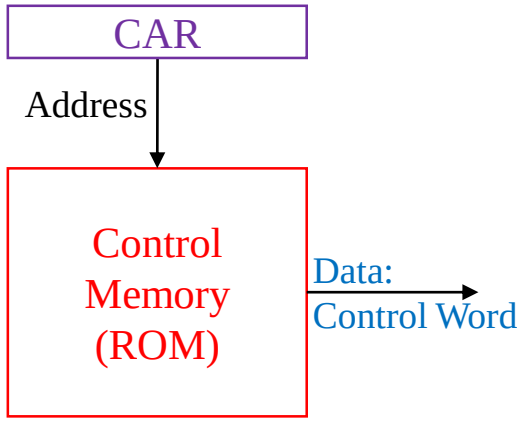
بنام خداوند جان و خرد

معماری کامپیوتر

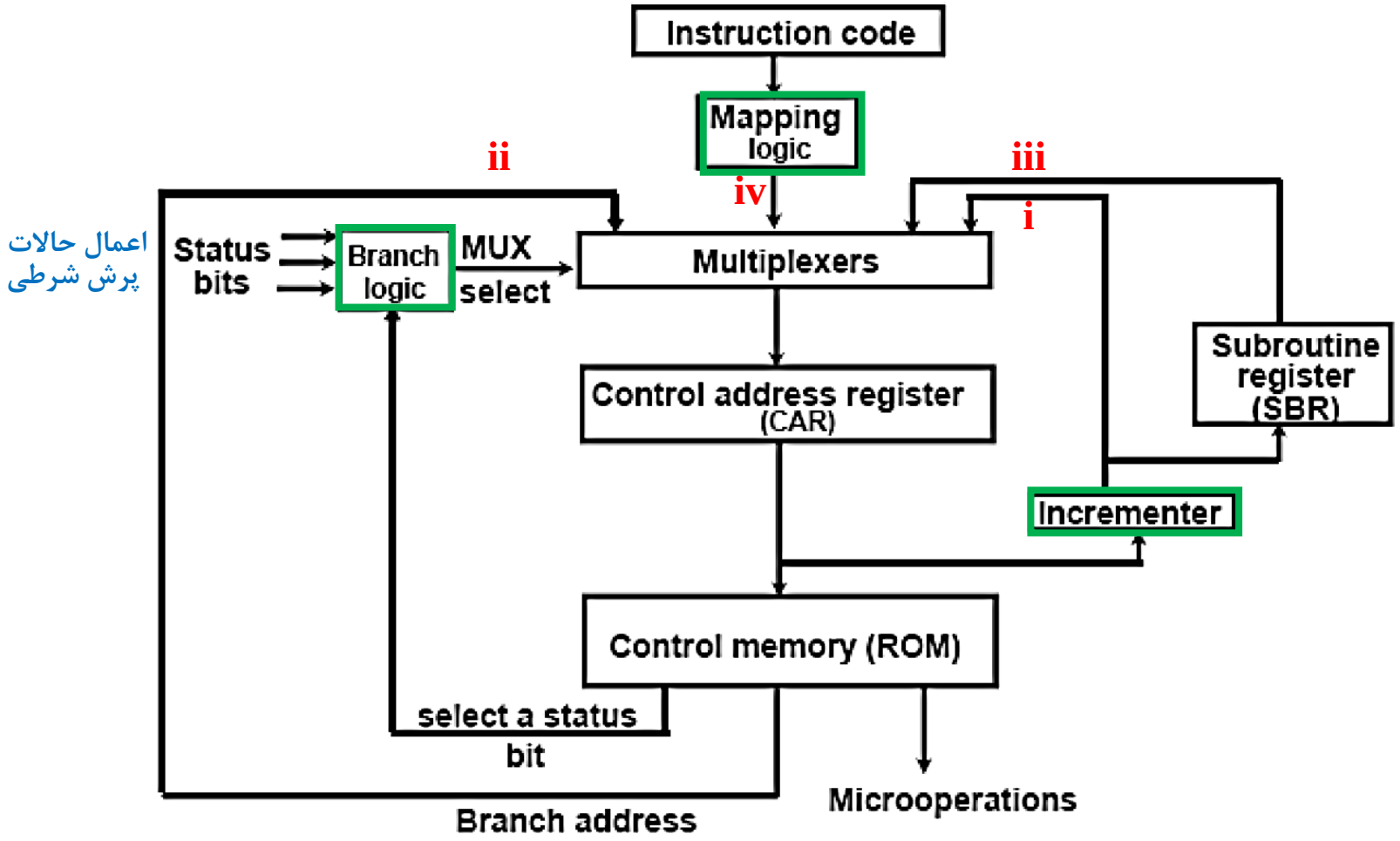
جلسه بیست و چهارم: پیاده‌سازی واحد کنترل (ریزبرنامه‌ریزی شده)

ساختار و روال عملکرد حافظه کنترلی

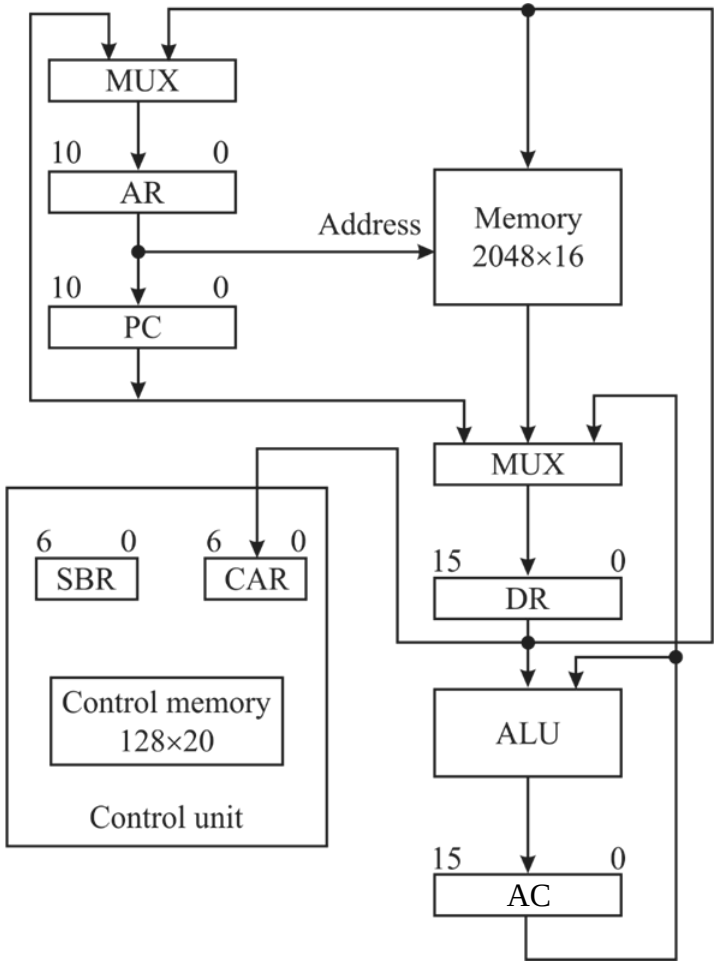
- از جنس ROM است و یک بخش آدرس و یک بخش داده دارد
- روال عملکرد:
  - آدرس ریزدستورالعمل جاری به حافظه کنترلی داده می‌شود (CAR)
  - به مکانی که آدرس مشخص شده رفته و داده آن را استخراج می‌شود (شیوه ذخیره داده مهم است)
  - قسمتی از داده که مربوط به اطلاعات کنترلی است در اختیار مسیر داده قرار می‌گیرد
  -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



واحد تولید آدرس بعدی



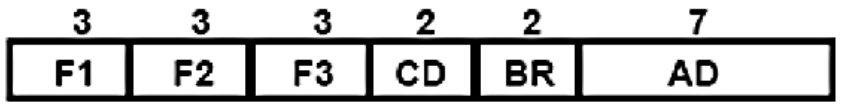
ذخیره داده در حافظه کنترلی



- ساختار پردازنده کامپیوتر نمونه چهاردستورالعملی:
- فرمت دستورالعمل‌ها:
  - ۱۱ بیت آدرس و ۴ بیت کدعملیاتی و ۱ بیت نوع آدرس‌دهی
- دستورات موردمطالعه:
  - Add, Branch, Store, Exchange

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



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

ذخیره داده در حافظه کنترلی-فیلدهای F

- نقطه اصلی طراحی ریزبرنامه، مشخص کردن ریزعملگرهاست
  - در طراحی سیم‌بندی شده امکان موازی‌سازی براساس منابع داشتیم
  - در این طراحی امکان موازی‌سازی به‌اندازه فیلدهای F داریم (حداکثر سه تا ریزدستورالعمل)
  - دستورات را به‌نحوی بین این سه فیلد تقسیم کرده‌اند که امکان اجرای موازی آن‌ها باشد
  - اگر از یک فیلد استفاده نکردیم، NOP در آن می‌گذاریم

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

F2	Microoperation	Symbol
000	None	NOP
001	AC ← AC - DR	SUB
010	AC ← AC ∨ DR	OR
011	AC ← AC ∧ DR	AND
100	DR ← M[AR]	READ
101	DR ← AC	ACTDR
110	DR ← DR + 1	INCDR
111	DR(0-10) ← PC	PCTDR

F3	Microoperation	Symbol
000	None	NOP
001	AC ← AC ⊕ DR	XOR
010	AC ← AC'	COM
011	AC ← shl AC	SHL
100	AC ← shr AC	SHR
101	PC ← PC + 1	INCPC
110	PC ← AR	ARTPC
111	Reserved	

ذخیره داده در حافظه کنترلی-فیلدهای F

- طراح (ریزبرنامه‌نویس) مشخص می‌کند چه ریزعملگرهایی امکان موازی‌سازی دارند
  - ریزعملگرها با مشخصات زیر مجاز هستند؟
    - F1:010, F2: 010,F3:000
    - F1:110, F2: 001, F3:000
    - F1:010, F2: 101, F3:101

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

F2	Microoperation	Symbol
000	None	NOP
001	AC ← AC - DR	SUB
010	AC ← AC ∨ DR	OR
011	AC ← AC ∧ DR	AND
100	DR ← M[AR]	READ
101	DR ← AC	ACTDR
110	DR ← DR + 1	INCDR
111	DR(0-10) ← PC	PCTDR

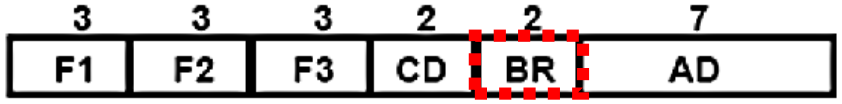
F3	Microoperation	Symbol
000	None	NOP
001	AC ← AC ⊕ DR	XOR
010	AC ← AC'	COM
011	AC ← shl AC	SHL
100	AC ← shr AC	SHR
101	PC ← PC + 1	INCPC
110	PC ← AR	ARTPC
111	Reserved	

ذخیره داده در حافظه کنترلی-فیلد CD

- شرط‌هایی که برای پرش می‌توانیم داشته باشیم
  - شرط‌هایی که رخداد پرش‌ها را معلوم می‌کنند
  - دوبیتی است پس چهار حالت می‌توان داشت
    - پرش غیرشرطی
    - شرط آدرس‌دهی غیرمستقیم (بیت I دستورالعمل)
    - شرط علامت AC
    - شرط صفر بودن AC

CD	Condition	Symbol	Comments
00	Always = 1	U	Unconditional branch
01	DR(15)	I	Indirect address bit
10	AC(15)	S	Sign bit of AC
11	AC = 0	Z	Zero value in AC

ذخیره داده در حافظه کنترلی-فیلد BR



- نوع پرش‌های ممکن را مشخص می‌کند و ۴ حالت است:
  - پرش ساده (JMP) به شرط CD
  - فراخوانی (Call)
  - بازگشت از یک فراخوانی (Ret)
  - نگاشت (Map)
    - تبدیل کدعملیاتی به آدرس (نگاشت بیتی)

BR	Symbol	Function
00	JMP	CAR ← AD if condition = 1
		CAR ← CAR + 1 if condition = 0
01	CALL	CAR ← AD, SBR ← CAR + 1 if condition = 1
		CAR ← CAR + 1 if condition = 0
10	RET	CAR ← SBR (Return from subroutine)
11	MAP	CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0

ذخیره داده در حافظه کنترلی

- ضرورت استفاده از حالت آدرس map چیست؟
  - این دستور، عمل نگاشت یک قالب دستورالعمل را در حافظه کنترلی انجام می‌دهد
  - این دستور برای عملیات fetch بکار می‌رود
  - آخر روتین fetch باید به دستورالعمل مربوطه پرش داشته باشیم
    - آدرس پرش مشخص نیست چون از ابتدا نمی‌دانیم چه دستوری در حال اجراست
    - پس با نگاشت در انتهای fetch به مکانی می‌رویم که روتین مربوط به opcode فعلی در آن ذخیره شده است
    - در نتیجه مستقل از نوع دستور، در یک گام از انتهای fetch به شروع اجرای دستور جاری می‌رویم

ذخیره داده در حافظه کنترلی

- برای اجرای ترتیبی و بدون پرش باید BR و CD را چگونه تنظیم کرد؟
  - AD آدرس پرش را مشخص می‌کند

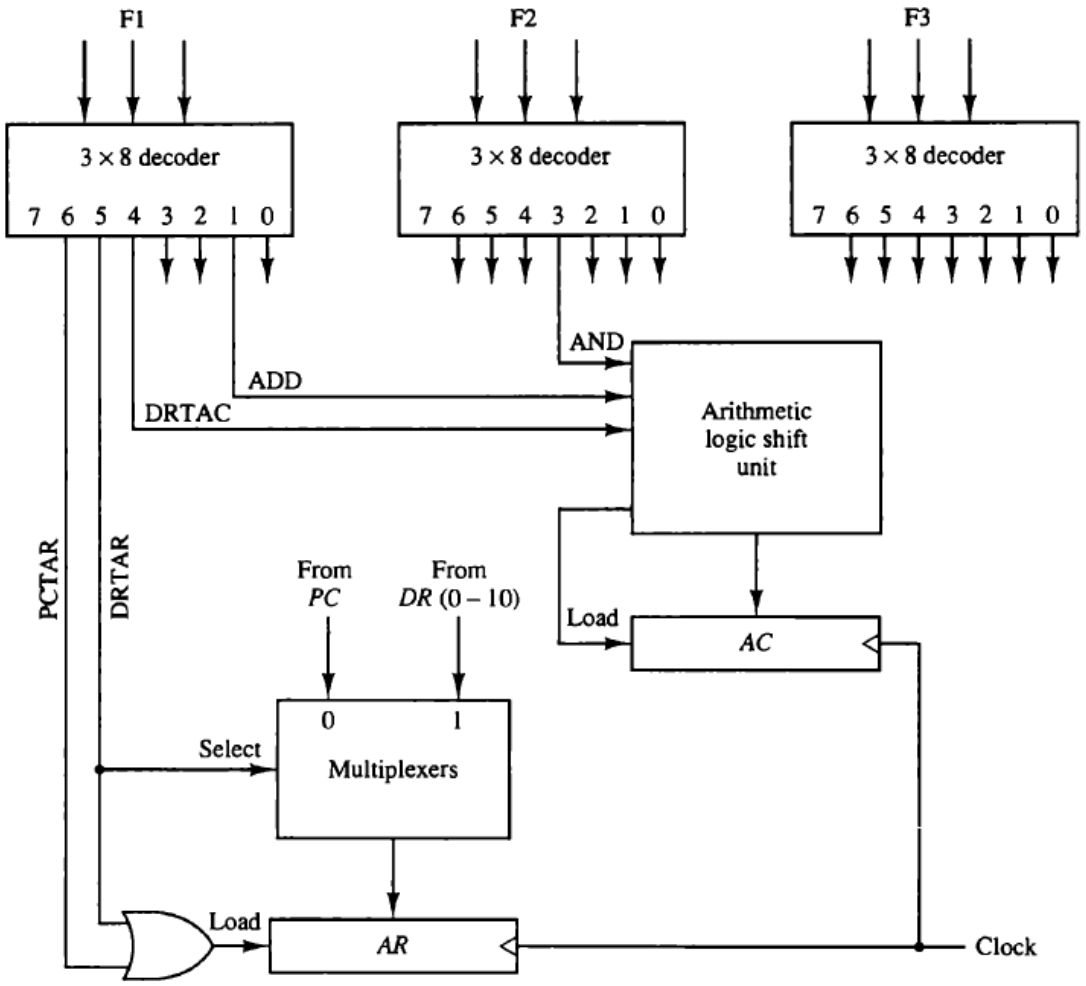
BR	Symbol	Function
00	JMP	CAR ← AD if condition = 1
		CAR ← CAR + 1 if condition = 0
01	CALL	CAR ← AD, SBR ← CAR + 1 if condition = 1
		CAR ← CAR + 1 if condition = 0
10	RET	CAR ← SBR (Return from subroutine)
11	MAP	CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0

CD	Condition	Symbol	Comments
00	Always = 1	U	Unconditional branch
01	DR(15)	I	Indirect address bit
10	AC(15)	S	Sign bit of AC
11	AC = 0	Z	Zero value in AC

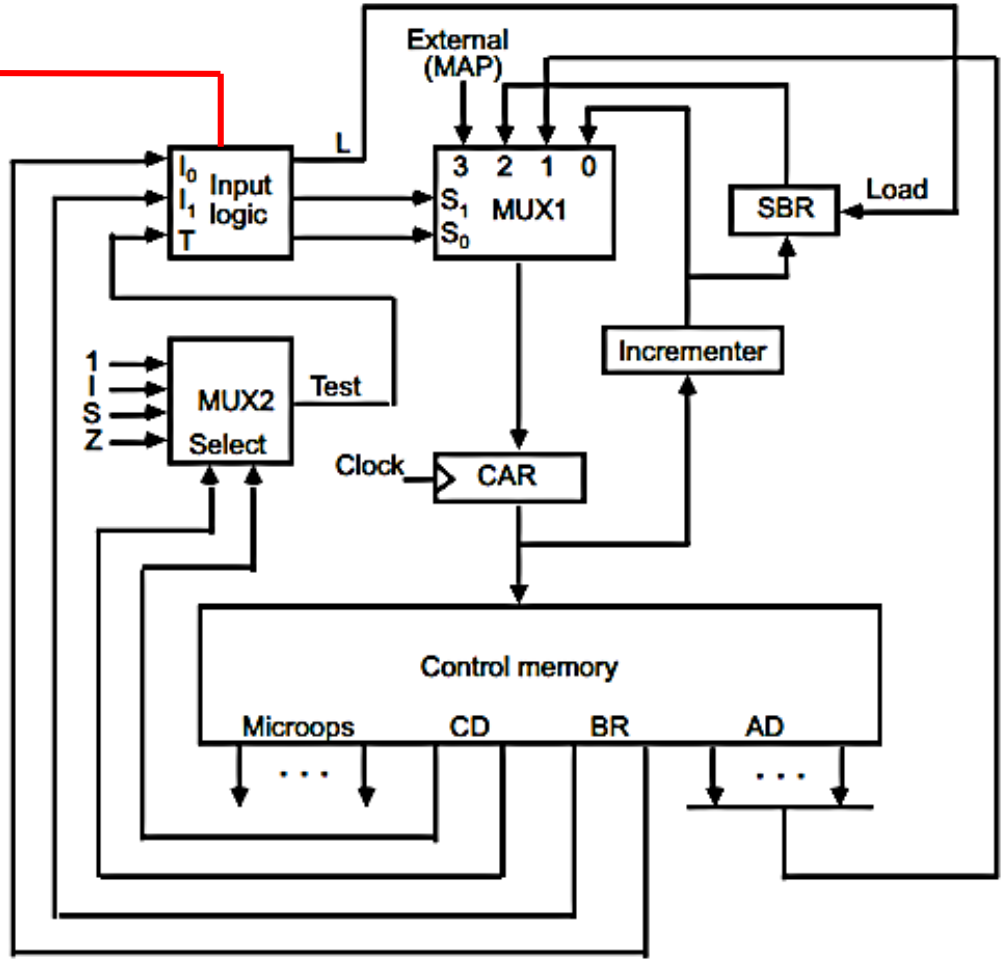
ساخت کلمه کنترلی از داده‌های حافظه کنترلی

- کلمات کنترلی از فیلدهای F ساخته می‌شوند
  - فیلدهای F توابع پیاده‌سازی شده برای اجرای دستور هستند
  - برای مشخص کردن کلمه کنترلی معادل هر فیلد F نیاز به یک ماژول دیکدر داریم
  - خروجی‌های این دیکدرها، به ورودی ثبات‌های مربوطه در مسیر داده متصل می‌شوند
  - اتصالات و طراحی همه اجزا به‌جز حافظه کنترلی به‌صورت سخت‌افزاری و سیم‌بندی شده می‌باشد

ساخت کلمه کنترلی از داده‌های حافظه کنترلی



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

$S_0 = I_0$

$S_1 = I_0I_1 + I_0'T$

$L = I_0'I_1T$