

برنامه نویسی به زبان ماشین و اسمبلی

برنامه نویسی سیستمی کامپیوترهای شخصی (۸۰۸۶)
(ویژه دانشجویان دوره کاردانی کامپیوتر دانشگاه فنی و حرفه‌ای)

تهیه و تألیف: آزاد نوری

سرشناسه	: نوری، آزاد، ۱۳۶۲ -
عنوان و نام پدیدآور	: برنامه‌نویسی به زبان ماشین و اسمبلی : برنامه نویسی سیستمی کامپیوترهای شخصی (۸۰۸۶) (ویژه دانشجویان دوره کاردانی کامپیوتر دانشگاه فنی و حرفه‌ای) / تهیه و تالیف آزاد نوری.
مشخصات نشر	: تهران : پندار پارس، ۱۳۹۴.
مشخصات ظاهری	: ۲۰۸ ص: مصور، جدول.
شابک	: 978-600-6529-95-0
وضعیت فهرست نویسی	: فیبا
یادداشت	: کتابنامه.
عنوان دیگر	: برنامه نویسی سیستمی کامپیوترهای شخصی (۸۰۸۶) (ویژه دانشجویان دوره کاردانی کامپیوتر دانشگاه فنی و حرفه‌ای).
موضوع	: اسمبلی (زبان برنامه نویسی کامپیوتر)
موضوع	: زبان‌های برنامه‌نویسی کامپیوتر
رده بندی کنگره	: ۷۳/۷۶QA الف/۱۳۹۴۵ ۱۳۹۳ن
رده بندی دیویی	: ۱۳۶/۰۰۵
شماره کتابشناسی ملی	: ۳۹۳۵۸۵۳

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگر جنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶

www.pendarepars.com info@pendarepars.com

تلفن: ۶۶۵۷۲۳۳۵ - تلفکس: ۶۶۹۲۶۵۷۸



نام کتاب	: برنامه نویسی به زبان ماشین و اسمبلی
ناشر	: انتشارات پندار پارس
گردآوری و تالیف	: آزاد نوری
چاپ نخست	: مهر ماه ۹۴
شمارگان	: ۱۰۰۰ نسخه
طرح جلد	: سارا یعسوبی
چاپ، صحافی	: روز
قیمت	: ۱۵۰۰۰ تومان
شابک	: ۹۷۸-۶۰۰-۶۵۲۹-۹۵-۰

*هرگونه کپی برداری، تکثیر و چاپ کاغذی یا الکترونیکی از این کتاب بدون اجازه ناشر تخلف بوده و پیگرد قانونی دارد *

فهرست مطالب

فهرست مطالب	۱
پیشگفتار	۱۱
فصل اول: سیستم اعداد	۱۳
مقدمه	۱۴
سیستم اعداد	۱۴
سیستم دودویی (باینری)	۱۴
تبدیل مبنای ۱۰ به ۲	۱۵
تبدیل مبنای ۲ به ۱۰	۱۶
سیستم اعداد هگزادسیمال (شانزده)	۱۷
تبدیل مبنای ۱۰ به ۱۶	۱۷
تبدیل مبنای ۱۶ به ۱۰	۱۸
تبدیل مبنای ۱۶ به ۲	۱۸
تبدیل مبنای ۲ به ۱۶	۱۸
مبنای ۸ (اوکتال)	۱۸
تبدیل مبنای ۱۰ به ۸	۱۹
تبدیل مبنای ۸ به ۱۰	۱۹
تبدیل مبنای ۸ به ۲	۱۹
تبدیل مبنای ۲ به ۸	۱۹
محاسبات در مبنای دودویی و هگزا دسیمال	۲۰
نحوه ذخیره سازی اعداد در حافظه	۲۲

۲۴	اعداد ده‌دهی BCD
۲۵	فصل دوم: معماری کامپیوتر
۲۶	مقدمه
۲۷	ویژگی‌های برنامه نویسی به زبان ماشین و زبان اسمبلی
۳۰	اجزاء متداول یک کامپیوتر
۳۰	واحد پردازنده یا CPU
۳۰	واحد ورودی/خروجی (I/O)
۳۱	واحد حافظه
۳۳	گذرگاه
۳۴	سیکل اجرای دستورات برنامه اسمبلی
۳۶	ارتباط با وسایل ورودی/خروجی
۳۶	بیت توازن (parity bit)
۳۶	ساختار پردازنده
۳۷	معرفی ثبات‌ها (رجیسترها) در پردازنده ۸۰۸۶
۳۸	ثبات‌های عمومی
۳۸	ثبات‌های اشاره گر و اندیس (شاخص)
۳۹	ثبات‌های سگمنت
۳۹	ثبات وضعیت/پرچم (PSW یا FR)
۴۲	ثبات‌های ۳۲ بیتی در پردازنده مدل ۸۰۳۸۶
۴۴	سگمنت (قطعه)

۴۶	نحوه تبدیل آدرس منطقی به فیزیکی.....
۴۷	انتقال اطلاعات
۴۹	حافظه پشته.....
۵۱	نحوه ترجمه برنامه توسط اسمبلر*.....
۵۳	کد اسکی (Ascii).....
۵۵	فصل سوم: مقدمات برنامه نویسی.....
۵۶	قوانین نام‌گذاری متغیرها.....
۵۶	قالب دستورات اسمبلی.....
۵۷	برچسب‌ها.....
۵۸	برچسب نام دستور.....
۵۹	برچسب نام متغیر.....
۵۹	تعریف متغیر.....
۶۱	تعریف آرایه.....
۶۱	روش‌های آدرس دهی اطلاعات و عملوندها.....
۶۱	آدرس دهی ضمنی.....
۶۲	آدرس دهی ثباتی.....
۶۲	روش آدرس دهی بلافاصله.....
۶۲	آدرس دهی مستقیم.....
۶۲	روش آدرس دهی غیرمستقیم.....
۶۳	آدرس دهی غیرمستقیم نسبی با ثبات پایه.....

۶۳	آدرس دهی غیرمستقیم نسبی با ثبات شاخص (اندیس).....
۶۴	آدرس دهی غیرمستقیم نسبی با ثبات پایه و شاخص.....
۶۴	شبه دستورات یا راهنماهای اسمبلر.....
۶۵	تعریف سگمنت.....
۶۶	تعریف رویه (روال).....
۶۸	قالب و ساختار برنامه اسمبلی.....
۶۸	قالب استاندارد.....
۶۹	برخی از شبه دستورات مهم قالب استاندارد.....
۶۹	شبه دستور PAGE.....
۶۹	شبه دستور TITLE.....
۶۹	شبه دستور ASSUME.....
۶۹	دستورات مقدار دهی اولیه ثبات‌های سگمنت.....
۷۰	شبه دستور END.....
۷۰	شبه دستورات بازگشت کنترل به سیستم عامل.....
۷۰	قالب ساده شده اسمبلی (Small).....
۷۱	قالب برنامه ساده شده (مدل small).....
۷۱	نمونه برنامه‌های آزمایشی در قالب‌های فوق.....
۷۳	نحوه اجرای برنامه اسمبلی.....
۷۷	برخی از شبه دستورات پر کاربرد.....
۷۷	شبه دستور DUP.....

۷۷	شبه دستور EQU (EQUATE).....
۷۸	شبه دستور ORG.....
۷۸	ثبات‌های افست آدرس (پیش فرض) برای ثبات‌های سگمنت.....
۸۱	فصل چهارم: دستورات عمل‌های اساسی در اسمبلی.....
۸۲	مقدمه.....
۸۲	انتقال داده‌ها (MOV).....
۸۶	اشاره گر.....
۸۶	OFFSET.....
۸۶	LEA.....
۸۷	LDS.....
۸۷	LES.....
۸۸	دستور XCHG.....
۸۹	INC.....
۸۹	DEC.....
۸۹	دستورات عمل محاسبه مکمل دو (NEG).....
۹۰	دستورات محاسباتی (جمع، تفریق، ضرب و تقسیم).....
۹۰	دستور جمع ADD.....
۹۱	دستور جمع با بیت نقلی ADC.....
۹۱	دستور تفریق SUB.....
۹۲	دستور تفریق با بیت فرضی SBB.....

۹۲ دستورات ضرب MUL و IMUL
۹۶ دستور PTR
۹۸ دستورات تقسیم DIV و IDIV
۱۰۴ مثال‌های برنامه نویسی
۱۱۱ فصل پنجم: ساختارهای انشعاب، تصمیم و حلقه‌های تکرار
۱۱۲ دستور مقایسه CMP
۱۱۳ دستورات انشعاب (پرش)
۱۱۳ پرش غیرشرطی
۱۱۴ پرش شرطی
۱۱۴ دستورات پرش شرطی مبتنی بر بیت پرچم
۱۱۵ دستورات پرش شرطی اعداد علامت دار
۱۱۶ دستورات پرش شرطی اعداد بدون علامت
۱۱۷ پیاده سازی ساختار IF-ELSE با دستورات پرش شرطی
۱۱۸ پیاده سازی ساختار SWITCH با دستورات پرش شرطی
۱۱۹ دستور پرش شرطی JCXZ
۱۲۰ حلقه‌های تکرار
۱۲۱ دستور LOOPD
۱۲۱ حلقه تکرار با دستورات پرش شرطی
۱۲۱ حلقه‌های تو در تو
۱۲۲ دستور LOOPZ یا LOOPE

۱۲۳	دستور LOOPNE یا LOOPNZ
۱۲۴	مثال‌های برنامه نویسی
۱۲۷	فصل ششم: عملیات منطقی و عملیات بیتی
۱۲۸	دستورات منطقی
۱۲۸	دستور NOT
۱۲۸	دستور AND
۱۲۹	دستور OR
۱۳۳	دستور XOR
۱۳۴	دستور TEST
۱۳۵	دستورات شیفت
۱۳۶	دستورات SHL و SAL
۱۳۷	دستور SHR
۱۳۷	دستور SAR
۱۳۸	دستورات چرخش
۱۳۹	دستور (Rotate Left) ROL
۱۳۹	دستور (Rotate Right) ROR
۱۴۰	دستور (Rotate Left Through Carry) RCL
۱۴۱	دستور (Rotate Right Through Carry) RCR
۱۴۲	دستورات کار با بیت‌های پرچم
۱۴۳	فصل هفتم: زیر برنامه، ماکرو و وقفه

۱۴۴ زیر برنامه (Subroutine)
۱۴۶ ساختار اصولی برنامه اسمبلی با چند روال
۱۴۹ ماکروها
۱۵۱ ماکروهای پارامتردار
۱۵۱ شبه دستور INCLUDE
۱۵۲ وقفه‌ها
۱۵۲ وقفه‌ها و عملیات ورودی/خروجی
۱۵۳ وقفه‌های سیستم
۱۵۳ مفهوم تابع وقفه
۱۵۴ نحوه اجرای وقفه‌ها
۱۵۴ مراحل اجرای وقفه
۱۵۵ دستور IRET
۱۵۵ وقفه‌های داخلی
۱۵۵ INT 00H (وقفه شماره صفر)
۱۵۵ INT 01H (وقفه شماره یک)
۱۵۵ INT 03H (وقفه شماره سه)
۱۵۶ دستور INTO (وقفه شماره چهار)
۱۵۶ معرفی توابع وقفه پر کاربرد در عملیات ورودی/خروجی
۱۵۶ پاک کردن مانیتور
۱۵۶ انتقال مکان نما

۱۵۷	خواندن کاراکتر از ورودی
۱۵۸	خواندن یک رشته از ورودی
۱۶۰	نمایش یک کاراکتر در خروجی (مانیتور)
۱۶۳	خواندن زمان و تاریخ سیستم
۱۶۵	فصل هشتم: رشته‌ها
۱۶۶	مقدمه
۱۶۶	رشته (STRING)
۱۶۷	تکرار اجرای دستورات پردازش رشته‌ای
۱۶۷	تعیین جهت پردازش رشته
۱۶۷	دستور MOVS (انتقال رشته‌ها)
۱۷۰	دستور LODS
۱۷۱	دستور STOS
۱۷۲	دستور CMPS (مقایسه رشته‌ها)
۱۷۵	دستور SCAS (جستجوی رشته)
۱۷۷	فصل نهم: برنامه‌های COM
۱۷۸	مقدمه
۱۷۸	ویژگی‌های برنامه‌های COM
۱۷۹	قالب و ساختار برنامه COM
۱۸۰	نحوه اجرای برنامه‌های COM به کمک توریو اسمبلر TASM
۱۸۳	فصل دهم: گرافیک در اسمبلی

۱۸۴	مقدمه
۱۸۴	صفحه نمایش و حافظه مانیتور
۱۸۶	کدهای کنترلی
۱۸۷	رنگ حروف و زمینه مانیتور در حالت متن
۱۸۹	حالت گرافیک مانیتور (Graphics Mode)
۱۹۰	تعیین حالت گرافیکی برای مانیتور (تابع 00 از وقفه INT 10H)
۱۹۱	روشن کردن پیکسل (تابع 0CH وقفه INT 10H)
۱۹۴	مثالهای تکمیلی
۱۹۷	فصل یازدهم: نرم افزار DEBUG
۱۹۸	نحوه اجرای برنامه DEBUG
۱۹۹	خلاصه‌ای از دستورات قابل اجرا در دیباگ
۲۰۰	نوشتن دستورات اسمبلی (ترجمه زبان اسمبلی به زبان ماشین)
۲۰۱	ترجمه محتویات حافظه (زبان ماشین) به زبان اسمبلی
۲۰۱	اجرای دستورات موجود در حافظه
۲۰۲	دستور اجرای برنامه اسمبلی به صورت دستور به دستور
۲۰۲	مشاهده محتویات ثبات‌ها
۲۰۴	فهرست منابع

پیشگفتار

زبان برنامه نویسی اسمبلی، زبان برنامه نویسی سخت افزار است و بیشترین نزدیکی را به زبان ماشین دارد. با توجه به مشکل بودن کار با زبان ماشین (صفر و یک)، برنامه نویسان از این زبان برای نوشتن برنامه‌های خود در کاربردهای مختلف سخت افزاری و صنعتی استفاده می‌کنند.

زبان ماشین و اسمبلی یکی از دروس مقطع کاردانی و کارشناسی دانشجویان رشته کامپیوتر است که با توجه به تفاوت‌های خاص آن با زبان‌های برنامه نویسی سطح بالا نیاز به دیدگاه و بینش متفاوت برای یادگیری آن وجود دارد. بر عکس زبان‌های سطح بالا نظیر ویژوال بیسک، سی شارپ و غیره که در آن‌ها نیاز به دانش قبلی در مورد ساختار سخت افزاری سیستم و اجزاء داخلی وجود ندارد، در زبان اسمبلی برنامه نویس باید ابتدا یک دید کلی از معماری سیستم و اجزاء داخلی پردازشگر پیدا کند و سپس با استفاده از امکانات داخلی سیستم و ویژگی‌های خاص هر واحد سخت افزاری اقدام به نوشتن برنامه خود کند.

کتاب‌های مرجع زیادی برای این درس وجود دارد که همگی آن‌ها به دلیل مرجع کامل بودن و حجیم بودن برای دانشجویان دوره کاردانی سنگین بوده و مناسب نیستند و علاوه بر این، بسیاری از این کتاب‌ها قدیمی بوده و مدت‌هاست که بروز رسانی نشده‌اند (بیش از ۱۰ الی ۱۵ سال است که مطالب آن‌ها بروز رسانی نشده است) لذا پس از تجربه چندین ساله تدریس این درس در دانشگاه‌های مختلف تصمیم بر این گرفته شد تا کتاب حاضر تهیه و در اختیار دانشجویان دوره کاردانی کامپیوتر دانشگاه فنی و حرفه‌ای قرار گیرد. کتاب به گونه‌ای تنظیم شده است که شامل تمامی سرفصل‌های مصوب وزارت علوم در دوره کاردانی بوده و ضمن پرهیز از ذکر مطالب اضافی و حجم بالا، تلاش شده است با زبانی ساده و روان مفاهیم تشریح شده و از مثال‌های مناسب و کافی استفاده شود. ضمناً تلاش شده است که با توجه به تغییرات سخت افزاری و خصوصاً تغییرات در نسل‌های جدید پردازنده‌ها از منابع جدیدتر و بروزتر استفاده شود.

این کتاب برای دانشجویان کاردانی و حتی کارشناسی سایر دانشگاه‌ها نیز قابل استفاده است.

از آنجا که زبان اسمبلی وابسته به ویژگی‌های سخت افزار است و از امکانات داخلی پردازشگر، حافظه و سایر قسمت‌ها استفاده می‌کند، لذا برنامه نویس بدون آشنایی با معماری کامپیوتر و اجزاء داخلی آن قادر به برنامه نویسی نخواهد بود. در بخش اول این کتاب مفاهیم

مقدماتی شامل سیستم اعداد و معماری کامپیوتر مورد بررسی قرار گرفته است و سپس در بخش دوم به آموزش مفاهیم برنامه نویسی زبان اسمبلی پرداخته شده است. در پایان لازم می دانم از زحمات آقای مهندس بهزاد نوری که در مراحل آماده سازی کتاب، یاری رسان بنده بودند تشکر نمایم.

فصل اول

سیستم اعداد

مقدمه

با توجه به اینکه زبان ماشین، زبان صفر و یک است و به عبارت دیگر سخت افزار از مبنای دودویی (Binary) استفاده می‌کند و از طرفی در محاسبات روزمره انسان‌ها، مبنای دهدهی (Decimal) مورد استفاده قرار می‌گیرد، لذا برنامه نویس باید با سیستم اعداد و نحوه تبدیل مبنای مختلف به یکدیگر آشنا باشد.

علاوه بر مبنای فوق در برنامه نویسی سخت افزار، برای راحت‌تر شدن نمایش اطلاعات و استفاده از تعداد رقم‌های کمتر در نمایش اطلاعات از مبنای شانزده (Hexadecimal) و بعضاً مبنای هشت (Octal) استفاده می‌شود. لذا برنامه نویس باید با نحوه نمایش اعداد در مبنای مختلف آشنا بوده و بتواند با اعداد در مبنای مختلف کار کند. در این فصل نحوه نمایش اعداد در مبنای مختلف، محاسبات در هر مبنای و همچنین نحوه تبدیل اعداد از یک مبنای دیگر توضیح داده شده است.

سیستم اعداد

نکته ۱: در مبنای r از ارقام ۰ تا $r-1$ استفاده می‌شود.

لذا در مبنای ۲ از ارقام ۰ و ۱ استفاده می‌شود و در مبنای ۱۰ از اعداد ۰ تا ۹ استفاده می‌گردد. همچنین در مبنای ۸ از ارقام ۰ تا ۷ و به همین ترتیب در مبنای ۱۶ از ارقام ۰ تا ۹ و کاراکترهای A تا Z (به جای ۱۰ تا ۱۵) استفاده می‌شود.

سیستم دودویی (باینری)

همانطور که قبلاً گفته شد در این مبنای ارقام ۰ و ۱ برای نمایش اطلاعات استفاده می‌شود. ارقام از راست به چپ شماره گذاری می‌شوند و هر کدام دارای ارزش مکانی خاصی هستند که برابر (شماره بیت ۲) است.

7	6	5	4	3	2	1	0	شماره بیت ها
*	*	*	*	*	*	*	*	عدد در مبنای دودویی
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	ارزش عددی
128	64	32	16	8	4	2	1	ارزش عددی دهدهی

شکل ۱-۱ سیستم دودویی (باینری)

علامت * در شکل فوق می تواند ۰ یا ۱ باشد. هر بیتی که ۱ باشد معادل ارزش عددی دهدهی خود را ایجاد می کند. نهایتاً مجموع ارزش های ایجاد شده معادل عدد دهدهی نمایش داده شده است.

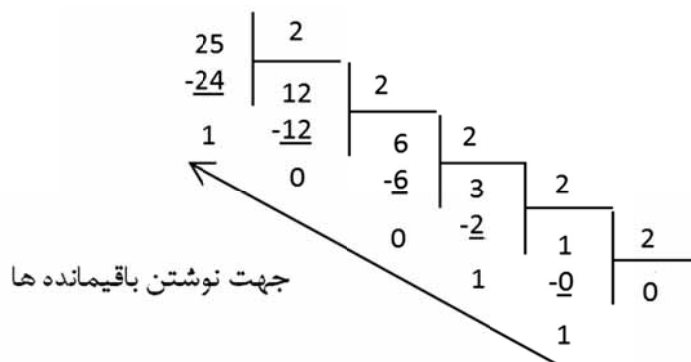
تبدیل مبنای ۱۰ به ۲

برای این کار به نکته زیر دقت کنید.

نکته ۲: برای تبدیل مبنای ۱۰ به هر مبنای دیگر (۲، ۸، ۱۶) کافی است عدد دهدهی را متوالیا بر مبنای مورد نظر تقسیم کرد و باقیمانده ها را از آخر به اول نوشت.

مطابق نکته فوق برای تبدیل عدد دهدهی به دودویی کافی است آن را متوالیا بر ۲ تقسیم کنیم تا زمانی که خارج قسمت صفر شود، آنگاه باقیمانده ها را از آخر به اول بنویسیم.

مثال: عدد ۲۵ معادل کدام عدد دودویی است؟



در نتیجه می توان نوشت:

$$25_{10} = 11001 = 0001\ 1001_2$$

اگر عدد دهدهی دارای قسمت اعشار نیز باشد، کافی است قسمت بعد از ممیز را متوالیا در عدد ۲ ضرب کنیم تا جاییکه دیگر رقمی پشت ممیز نماند. سپس اعداد صحیح به دست آمده را از بالا به پایین می نویسیم.

به عنوان نمونه فرض کنید می خواهیم عدد $۲۵/۱۲۵$ را به مبنای ۲ ببریم:

قسمت صحیح عدد که ۲۵ است را مانند روش مثال قبل به طور جداگانه محاسبه می کنیم. (تقسیم

$$25 = 11001 \quad (\text{متوالی بر } ۲)$$

اما برای قسمت اعشار (بعد از ممیز) که $۰/۱۲۵$ است داریم:

$$\begin{array}{l} 0.125 * 2 = \underline{0.250} \\ 0.250 * 2 = \underline{0.500} \\ 0.500 * 2 = \underline{1.000} \end{array} \quad \downarrow \quad \begin{array}{l} \text{از بالا به پایین فقط قسمت‌های صحیح را می‌نویسیم} \\ 25.125_{10} = 11001.001_2 \end{array}$$

مثال: عدد 25.875 را به معادل باینری آن تبدیل کنید.

قسمت صحیح قبلاً محاسبه شده است و فقط قسمت اعشاری را محاسبه می‌کنیم.

$$\begin{array}{l} 0.875 * 2 = 1.750 \\ 0.750 * 2 = 1.500 \\ 0.500 * 2 = 1.000 \end{array} \quad \downarrow \quad 0.875 = 111 \Rightarrow 25.875_{10} = 11001.111_2$$

نکته ۳: برای تبدیل مبنای ۲، ۸ و ۱۶ به مبنای ۱۰، ارزش مکانی ایجاد شده هر رقم را محاسبه و نتایج را با هم جمع می‌کنیم.
 ارزش مکانی هر رقم در هر مبنایی با رابطه زیر محاسبه می‌شود: $\text{رقم} * \text{شماره رقم (مبنا)}$

تبدیل مبنای ۲ به ۱۰

مطابق نکته ۳، برای این کار ابتدا بیت‌ها را شماره گذاری می‌کنیم. سپس به ازای بیت‌هایی که مقدار ۱ دارند، ارزش مکانی (وزنی) هر بیت (شماره بیت)، را محاسبه کرده و نتایج را با هم جمع می‌کنیم.

مثال: عدد ۱۰۱۱۰۱/۱۰۱ را به معادل ده‌دهی تبدیل کنید.

$$\begin{array}{cccccccc} 5 & 4 & 3 & 2 & 1 & 0 & -1 & -2 & -3 \\ 1 & 0 & 1 & 1 & 0 & 1 & . & 1 & 0 & 1 \end{array} \quad \text{فقط بیت‌هایی را که مقدار ۱ دارند در نظر می‌گیریم}$$

$$1 * 2^5 + 1 * 2^3 + 1 * 2^2 + 1 * 2^0 + 1 * 2^{-1} + 1 * 2^{-3} = 32 + 8 + 4 + 1 + 0.5 + 0.125 = 45.625$$

$$\rightarrow 101101.101_2 = 45.625_{10}$$

(مبنای ۸ و ۱۶ نیز به همین شکل و مطابق نکته ۳ به مبنای ۱۰ تبدیل می‌شوند. در ادامه همین فصل خواهیم دید)

حتی می‌توان خود اعداد ده‌دهی را نیز به همین روش نمایش داد به عنوان مثال:

$$\begin{aligned} 123.45 &= 1 * 10^2 + 2 * 10^1 + 3 * 10^0 + 4 * 10^{-1} + 5 * 10^{-2} \\ &= 100 + 20 + 3 + 0.4 + 0.05 \end{aligned}$$

سیستم اعداد هگزادسیمال (شانزده)

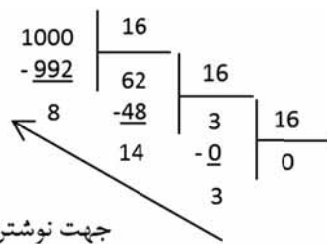
در این مبنا برای نمایش اعداد از ارقام ۰ تا ۹ و کاراکترهای A تا F استفاده می‌شود. در جدول ۱-۱ می‌توانید ارقام هگزا دسیمال و معادل دهدهی و باینری آنها را ببینید. هر رقم هگزا دسیمال معادل ۴ بیت باینری است.

جدول ۱-۱ اعداد هگزادسیمال

ارقام هگزا دسیمال	معادل باینری (۲)	معادل دسیمال (۱۰)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

تبدیل مبنای ۱۰ به ۱۶

مطابق نکته ۲ در همین فصل کافی است عدد دهدهی را متوالیا بر ۱۶ تقسیم کنیم تا جائیکه به صفر برسیم آنگاه باقیمانده‌ها را از آخر به اول بنویسیم. فقط باید دقت کرد که به جای اعداد ۱۰ تا ۱۵ از کاراکتر معادل مطابق جدول ۱-۱ استفاده شود.



جهت نوشتن باقیمانده‌ها

مثلا فرض کنید می‌خواهیم عدد ۱۰۰۰ را به مبنای ۱۶ ببریم.

لذا داریم : $1000_{10} = 3E8_{16}$

تبدیل مبنای ۱۶ به ۱۰

مطابق نکته شماره ۳ باید ارقام را شماره گذاری و پس از محاسبه ارزش مکانی ارقام، آن‌ها را با هم جمع کرد.

مثال: عدد 3E8 را به مبنای ۱۰ ببرید.

$$\begin{array}{r} 2 \quad 1 \quad 0 \\ 3E8 \end{array} = 3 \cdot 16^2 + E \cdot 16^1 + 8 \cdot 16^0 = 3 \cdot 256 + 14 \cdot 16 + 8 \cdot 1 = 1000$$

تبدیل مبنای ۱۶ به ۲

هر رقم هگزادسیمال معادل ۴ رقم (بیت) باینری است لذا کافی است هر رقم هگز را در ۴ بیت نمایش داده و به ترتیب پشت سرهم نوشت.

مثال: عدد A2B را به مبنای دو (باینری) تبدیل کنید.

$$\begin{array}{ccc} & A & 2 & B \\ & \swarrow & \downarrow & \searrow \\ 1010 & 0010 & 1011 & \end{array} \quad \rightarrow \quad A2B = 1010 \ 0010 \ 1011$$

تبدیل مبنای ۲ به ۱۶

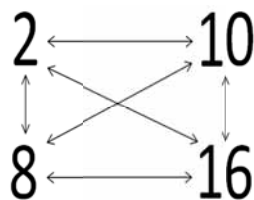
برای این کار کافی است از راست به چپ ۴ رقم ۴ رقم بیت‌ها را جدا نموده و به ازای هر ۴ رقم (بیت) معادل هگز آن را نوشت.

مثال: عدد ۰۱۰۰۱۱۱۱۰۰۱۱ به مبنای ۱۶ ببرید.

$$\begin{array}{ccc} 0100 & '1111' & 0011 \\ \downarrow & \downarrow & \downarrow \\ 4 & F & 3 \end{array}$$

نکته ۴: با تغییر مبنا فقط نحوه نمایش هر عدد تغییر می‌یابد اما مقدار ارزش آن تغییر نخواهد کرد

چنانچه در شکل روبرو می‌بینید برای تبدیل مبناها به یکدیگر می‌توان مستقیم یا غیر مستقیم اقدام



کرد مثلا تبدیل ۱۰ به ۱۶ هم مستقیما انجام پذیر است و هم از طریق تبدیل ۱۰ به ۲ و سپس ۲ به ۱۶ یا تبدیل ۱۰ به ۸ و سپس ۸ به ۱۶.

مبنای ۸ (اوکتال)

در این مبنا از اعداد ۰ تا ۷ برای نمایش اعداد استفاده می‌شود. هر رقم اوکتال معادل ۳ بیت (رقم) باینری است.

تبدیل مبنای ۱۰ به ۸

مطابق نکته ۲ کافی است عدد دهدهی را متوالیا بر ۸ تقسیم نموده و باقیمانده‌ها را از آخر به اول نوشت.

مثال: عدد ۲۲ را به مبنای ۸ تبدیل کنید.

$$\begin{array}{r|l} 22 & 8 \\ -16 & 2 \\ \hline 6 & -0 \\ & 2 \end{array}$$

در نتیجه داریم: $22_{10} = 26_8$

جهت نوشتن باقیمانده‌ها

تبدیل مبنای ۸ به ۱۰

مطابق نکته شماره ۳ باید ارقام را شماره گذاری و پس از محاسبه ارزش مکانی ارقام، آن‌ها را با هم جمع کرد.

مثال: عدد ۲۶ در مبنای ۸ را به مبنای ۱۰ ببرید.

$$\begin{array}{cc} 1 & 0 \\ 26 & \end{array}$$

$$2 \cdot 8^1 + 6 \cdot 8^0 = 16 + 6 = 22$$

تبدیل مبنای ۸ به ۲

هر رقم اوکتال (هشت) معادل ۳ بیت باینری است. لذا کافی است به ازای هر رقم معادل باینری آن را در ۳ بیت نوشت.

مثال: عدد ۵۳ اوکتال را به باینری تبدیل کنید.

$$\begin{array}{cc} 5 & 3 \\ \swarrow & \searrow \\ 101 & 011 \end{array}$$

$$53_o = 101\ 011_B$$

تبدیل مبنای ۲ به ۸

برای اینکار کافی است عدد مورد نظر را از راست به چپ ۳ رقم ۳ رقم جدا نموده و به ازای هر ۳ رقم، معادل اوکتال آن را بنویسید.

مثال: عدد ۱۱۰۰۱ را به معادل مبنای ۸ آن تبدیل کنید.

011'001
↓ ↓
3 1

پس می‌توان نوشت:

$011001_8 = 31_{10}$ یا به عبارت دیگر $011001_2 = 31_8$

محاسبات در مبنای دودویی و هگزا دسیمال

عملیات جمع و تفریق در مبنای ۲ و ۱۶ از همان قوانین مبنای ۱۰ تبعیت می‌کند. با این تفاوت که به جای عدد ۱۰ مبنای مورد نظر (۲ یا ۱۶) را مد نظر قرار می‌دهیم. مثلاً در عملیات مبنای ۱۰ (دهه‌ای) داریم:

عملیات جمع: اگر مجموع ارقام یک ستون بزرگ‌تر یا مساوی ۱۰ باشد، ۱۰ واحد را از آن کم نموده و باقیمانده را می‌نویسیم در ضمن ۱ واحد به ارقام ستون سمت چپ اضافه می‌کنیم.
عملیات تفریق: اگر مقدار رقم اول (مفروق منه) کمتر از مقدار رقم دوم (مفروق) باشد، از ستون سمت چپ آن ۱ واحد کم نموده و به ازای آن ۱۰ واحد به ستون سمت راست اضافه می‌کنیم و سپس عملیات تفریق را انجام می‌دهیم.

اکنون همین قوانین برای سایر مبنایها نیز صادق است فقط کافی است به جای عدد ۱۰، عدد مبنای مورد نظر (۲ یا ۱۶) را قرار دهیم.

پس در مبنای ۲ داریم:

عملیات جمع: اگر مجموع ارقام یک ستون بزرگ‌تر یا مساوی ۲ باشد، ۲ واحد را از آن کم نموده و باقیمانده را می‌نویسیم در ضمن ۱ واحد به ارقام ستون سمت چپ اضافه می‌کنیم.
عملیات تفریق: اگر مقدار رقم اول (مفروق منه) کمتر از مقدار رقم دوم (مفروق) باشد، از ستون سمت چپ آن ۱ واحد کم نموده و به ازای آن ۱۰ واحد به ستون سمت راست اضافه می‌کنیم و سپس عملیات تفریق را انجام می‌دهیم.

همچنین می‌توان از جدول ۱-۲ برای عملیات جمع استفاده کرد.

جدول ۱-۲ عملیات جمع در مبنای باینری

	SUM(مجموع)	CARRY(رقم نقلی)
0+0	0	0
0+1	1	0
1+0	1	0
1+1	0	1
1+1+1	1	1

مثال: عملیات جمع زیر را در سیستم باینری انجام دهید.

$$\begin{array}{r}
 \overset{19}{0001} \overset{1}{00} \overset{1}{11} \\
 + \overset{7}{0000} 0111 \\
 \hline
 0001 1010
 \end{array}$$

19+7

سخت افزار برای انجام تفریق دودویی از همان مدار جمع کننده استفاده می کند یعنی با استفاده از رابطه زیر ابتدا عملیات تفریق را به جمع تبدیل کرده و سپس محاسبه را انجام می دهد.

A-B=A+(B مکمل دو عدد)

از طرفی داریم: مکمل دو = مکمل یک + ۱ و همچنین مکمل یک از تبدیل ۰ به ۱ و ۱ به ۰ به دست می آید.

به عنوان مثال فرض کنید می خواهیم عملیات ۷-۱۹ را در سیستم باینری انجام دهیم.

$$\begin{array}{r}
 0000 0111 \quad 7 \\
 \\
 \begin{array}{r}
 1111 1000 \quad 7 \text{ مکمل } 1 \text{ عدد } 7 \\
 + \quad \quad \quad 1 \\
 \hline
 1111 1001 \quad 7 \text{ مکمل } 2 \text{ عدد } 7
 \end{array}
 \end{array}$$

$$\begin{array}{r}
 \overset{1}{1} \overset{1}{1} \overset{1}{1} \overset{1}{1} \overset{1}{1} \\
 0001 0011 \quad 19 \\
 + \quad \quad \quad 1111 1001 \quad 7 \text{ مکمل } 2 \text{ عدد } 7 \\
 \hline
 1 \quad 0000 1100
 \end{array}$$

مطابق نکات گفته شده در ابتدای این بخش در مبنای ۱۶ داریم:

عملیات جمع: اگر مجموع ارقام یک ستون بزرگتر یا مساوی ۱۶ باشد، ۱۶ واحد را از آن کم نموده و باقیمانده را می نویسیم در ضمن ۱ واحد به ارقام سمت چپ اضافه می کنیم.

عملیات تفریق: اگر مقدار رقم اول (مفروق منه) کمتر از مقدار رقم دوم (مفروق) باشد، از ستون سمت چپ آن ۱ واحد کم نموده و به ازای آن ۱۶ واحد به ستون سمت راست اضافه می‌کنیم و سپس عملیات تفریق را انجام می‌دهیم.

در مثال‌های زیر جمع و تفریق در مبنای هگزا دسیمال را می‌بینید.

جمع دو عدد

$$\begin{array}{r}
 2ABA \\
 + 1129 \\
 \hline
 3BE3
 \end{array}$$

$10+9=19 > 16$
 $19-16=3$

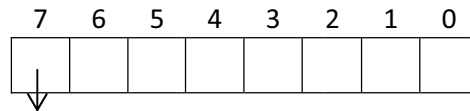
تفریق دو عدد

$$\begin{array}{r}
 2A1 \\
 - 115 \\
 \hline
 18C
 \end{array}$$

$1 < 5 \Rightarrow 1+16=17$
 $17-5=12$

نحوه ذخیره سازی اعداد در حافظه

اعداد در حافظه به دو شکل علامت دار و بدون علامت قابل ذخیره سازی است. در حالت علامت‌دار آخرین بیت به عنوان بیت علامت^۱ در نظر گرفته می‌شود که اگر ۰ باشد به معنای عدد مثبت و اگر ۱ باشد به معنای عدد منفی است. در شکل زیر ساختار یک بایت حافظه را در حالت علامت‌دار می‌بینید.



بیت علامت

در حالت علامت‌دار می‌توان در یک بایت از عدد ۱۲۸- تا ۱۲۷+ را ذخیره کرد اما اگر اعداد بدون علامت ذخیره شوند می‌توان در یک بایت حافظه، از ۰ تا ۲۵۵ را ذخیره نمود.

^۱ Sign Bit

نکته ۵: با n بیت می توان 2^n عدد یا حالت منحصر بفرد (غیر تکراری) را نمایش داد. این اعداد از 0 تا $2^n - 1$ است. به عبارت دیگر بزرگترین عددی که در n بیت قابل نمایش است، $2^n - 1$ است. (در حالت بدون علامت)

به عنوان مثال با ۲ بیت می توان ۴ عدد مختلف (۰ تا $2^2 - 1$) را نمایش داد: ۰۰، ۰۱، ۱۰، ۱۱
 با توجه به نکته فوق، در حالت بدون علامت، بزرگترین عددی که می توان در یک بیت (هشت بیت) نمایش داد معادل $2^8 - 1 = 255$ است و این عدد زمانی حاصل می شود که تمام بیت ها ۱ باشند (۱۱۱۱ ۱۱۱۱) و کوچکترین عدد قابل نمایش ۰ است که مربوط به زمانی است که همه بیت ها ۰ باشند (۰۰۰۰ ۰۰۰۰). به عبارت دیگر از 00 تا FF هگز قابل نمایش است.

نکته ۶: اعداد منفی به شکل مکمل ۲ در حافظه ذخیره می شوند.

به عنوان مثال عدد ۲۰- اینگونه در حافظه ذخیره می شود.

عدد ۲۰: ۰۰۰۱ ۰۱۰۰ اکنون از این عدد مکمل دو گرفته می شود.

مکمل دو عدد ۲۰: $\underline{1110 1100}$ نشان دهنده منفی بودن عدد

مثال: عدد علامت دار ۱۰۱۰ ۱۱۱۱ که در حافظه ذخیره شده است معادل کدام عدد دهدهی است؟

روش اول: با توجه به اینکه بیت علامت ۱ است پس می توان نتیجه گرفت که عدد منفی است و

به شکل مکمل ۲ در حافظه ذخیره شده است. اکنون کافی است مجدداً از آن مکمل ۲ بگیریم چون

داریم: مکمل دو (مکمل دو عدد x) برابر است با خود عدد x .

پس از گرفتن مکمل دو به دست می آید: ۰۰۰۰ ۰۱۱۰ و پس از تبدیل به مبنای ۱۰ عدد ۶- به

دست می آید.

روش دوم:

با توجه به بیت علامت و منفی بودن عدد، به جای گرفتن مکمل دو کافی است ابتدا معادل دهدهی

عدد فوق (۱۱۱۱ ۱۰۱۰) را به دست آوریم و سپس آن را منهای 2^8 یعنی ۲۵۶ کنیم.

$$1111 1010 = 128+64+32+16+8+2=250$$

$$250-256=-6$$

همانگونه که می بینید عدد ۶- به دست آمد که معادل نتیجه روش قبلی است.

مثال: عدد علامت دار ۰۱۰۱۰۰۰۰ معادل کدام عدد دهدهی است؟

با توجه به بیت علامت این عدد مثبت است، پس مستقیماً به دهدهی تبدیل می‌شود که عدد +۵ به دست می‌آید.

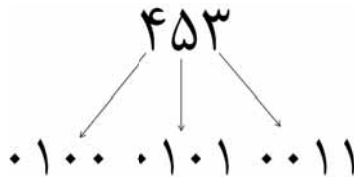
در جدول ۱-۳ محدوده اعداد قابل نمایش در یک و دو بایت در مبنای دهدهی و هگزادسیمال قابل مشاهده است.

جدول ۱-۳ محدوده اعداد قابل نمایش در حافظه

یک بایت		دو بایت		حالت ذخیره سازی
بدون علامت	علامت دار	بدون علامت	علامت دار	
۰ تا ۲۵۵	+۱۲۷ تا -۱۲۸	۰ تا ۶۵۵۳۵	+۳۲۷۶۷ تا -۳۲۷۶۸	محدوده اعداد قابل نمایش
00 تا FF		0000 تا FFFF		محدوده اعداد قابل نمایش (هگز)

اعداد دهدهی BCD^۱

برای سهولت انتقال اطلاعات از ورودی به حافظه و از حافظه به خروجی و همچنین سهولت در نمایش اعداد دهدهی، از روش BCD استفاده می‌شود. در این روش هر رقم در ۴ بیت نمایش داده می‌شود. به عنوان مثال برای تبدیل عدد BCD ۴۵۳ کافی است ارقام را از هم جدا و هر رقم را در ۴ بیت باینری نمایش دهیم.



برای تبدیل کد باینری به دهدهی BCD کافی است از راست به چپ به ۴ بیت ۴ بیت جدا نموده و معادل آن را بنویسیم. عدد BCD به دو صورت BCD فشرده و غیر فشرده به کار می‌رود. در حالت BCD فشرده در هر بایت می‌توان دو رقم BCD را ذخیره کرد در حالیکه در BCD غیر فشرده در هر بایت یک رقم ذخیره می‌شود (در نیمه سمت راست) و در نیمه سمت چپ صفر قرار می‌گیرد.

^۱ Binary Coded Decimal

فصل دوم

معماری کامپیوتر