نمایش داده ها

اکثر ساختمان های داده انتزاعی هستند که توسط برنامه نویس با مجموعه ای از دستورالعمل ها تعریف می شوند. نوع های داده پایه (نظیر اعداد باینری صحیح یا ممیز شناور، رشته های بیتی، کاراکترها و غیره (مستقیما در سخت افزار همراه با مجموعه ای از دستورالعمل طراحی می شوند. یک برنامه نویس زبان اسمبلی باید بداند چگونه سخت افزار این انواع داده های اصلی را پیاده سازی می کند .

واحدهای اطلاعاتی
روش های نمایش داده ها
نمایش اعداد صحیح
کدگذاریASCII
کاهش و افزایش طول داده

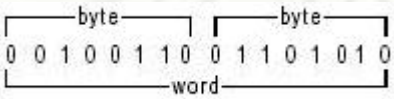
واحدهای اطلاعاتی

هر رقم در یک عدد باینری یک **بیت** (bit) نامیده می شود. بیت کوچکترین واحد اطلاعاتی در کامپیوتر است .

بیت ها به گروه های بزرگتری سازماندهی می شوند؛ در کامپیوترهای امروزی هر هشت بیت یک **بایت** (Byte) درنظر گرفته می شود که کوچکترین مکان آدرس پذیر حافظه است (که می تواند متفاوت از مقدار حافظه واکشی شده درهربار مراجعه باشد). یک بایت می تواند حاوی یک دستورالعمل ماشین، یک کاراکتر، یا یک عدد باشد .

یک نیبل (nibble) نیمه یک بایت یا چهار بیت است.

نوع بزرگتر ذخیره سازی یک **کلمه** (word) است که روی پردازنده های اینتل ۲ بایت (۱۶ بیت) است. یک کلمه طول پیش فرض داده است که توسط طراح پردازنده انتخاب شده است و منعکس کننده برخی نکات سخت افزاری نظیر گذرگاه های درونی و بیرونی است. کامپیوترهای شخصی اولیه با پردازنده های اینتل دارای عملوندهای ۱۶ بیتی بودند به همین دلیل کلمه به صورت ۱۶ بیتی تعریف شد. در پردازنده های دیگر طول کلمه الزاما ۲ بایت نیست .



یک **کلمه مضاعف** (doubleword) چهار بایت یا ۳۲ بیت طول دارد و یک **کلمه چهارگانه** (quadword) دارای هشت بایت یا ٦٤ بیت است .

endian

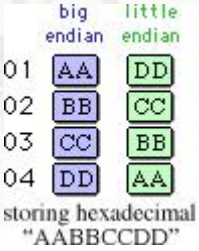
یک نکته مهم دیگر بعد از تعداد بیت ها ترتیب قرار گیری بایت های داده در حافظه است که باید موردتوجه برنامه نویس اسمبلی باشد. درحینی که پردازنده به طور نامحسوس endian را استفاده می کند، برای دسترسی به داده های چندبایتی حافظه و کارکردن با هر بایت بطور جداگانه، دانستن endian حیاتی است .ترتیب قرار گیری بایت ها در حافظه را برای داده های چندبایتی معین می کند. فرمت های زیر برای ذخیره یک مقدار چندبایتی وجود دارد :

۱. Big-endian
- داده ها را به ترتیب طبیعی خودشان ذخیره می کند. بایت با ارزش در کمترین آدرس قرار می گیرد. اکثر پردازنده های RISCو Motorola 68300 از این دسته هستند .

۲. Little-endian
- بایت با ارزش کمتر در آدرس های پائین تر حافظه ذخیره می شود. پردازنده های Intel x86 و Pentium از این نمونه هستند.

۳. Bi-endian
- پردازنده هائی مانند Motorola/IBM PowerPC می تواند در مد big-endian یا little-endian تحت کنترل نرم افزار کار کند .

مثال. داده چهار بایتی هگز AABBCCDD را درنظر بگیرید. نحوه تخصیص حافظه به این داده در دو فرمت endian به صورت زیر نشان داده شده است .مشاهده می شود که این دو فرمت عکس یکدیگر هستند .



روش های نمایش داده ها

اطلاعات معمولا به دو صورت استفاده می شوند: داده عددی (صحیح و ممیزشناور) و داده حرفی. نحوه نگهداری اطلاعات در حافظه را نمایش داده می گویند .روش های نگهداری داده ها بسته به نوع آنها متفاوت است .

نمایش اعداد صحیح(Integer)

اعداد صحیح باینری به دو شکل دیده می شوند :

۱. اعداد صحیح بدون علامت (unsigned Integer) که شامل اعداد صحیح غیر منفی هستند.
۲. اعداد صحیح علامت دار (signed Integer) که می توانند مثبت یا منفی باشند.

نمایش اعداد صحیح بدون علامت

در اعداد صحیح بدون علامت کلیه بیت ها به داده اختصاص داده می شود .کمترین مقدار ممکن یک عدد صحیح بدون علامت وقتی است که کلیه بیت ها صفر باشد که معادل عدد 0 است. در بزرگترین عدد صحیح بدون علامت کلیه بیت های عدد یک است .

نمایش اعداد صحیح علامت دار

اعداد صحیح علامت دار ممکن است مثبت یا منفی باشند. برای تشخیص علامت عدد یکی از بیت ها را به بیت علامت اختصاص می دهند. سه تکنیک برای نمایش علامت عدد وجود دارد که در نمایش اعداد صحیح علامت دار در حافظه استفاده می شده اند. در کلیه این روش ها با ارزش ترین بیت (سمت چپ ترین بیت) را به عنوان بیت علامت (sign bit) درنظر می گیرند. اگر این بیت ۰ باشد عدد مثبت و اگر ۱ باشد عدد منفی است .

روش های نمایش اعداد صحیح علامت دار عبارتند از:

۱. علامت مقدار
۲. مکمل ۱
۳. مکمل ۲

روش نمایش مکمل۲

دو روش اول در کامپیوترهای اولیه به کار می رفتند. کامپیوترهای امروزی روش سوم به نام نمایش مکمل ۲ را استفاده می کنند. مکمل۲ یک عدد در دو مرحله بدست می آید :

۱. پیداکردن مکمل۱ عدد
۲. اضافه کردن ۱ واحد به نتیجه مرحله اول

مثال. برای بدست آوردن مکمل۲ عدد 56 ابتدا مکمل۱ آن محاسبه سپس یک واحد به آن اضافه می شود .

56 = 00111000b
1's complement = 11000111
2's Complement = 11000111+1
=11001000

سیستم مکمل۲ روش خوبی برای ذخیره اعداد صحیح علامت دار به صورت باینری است. طول نمایش یا تعداد بیت هایی که استفاده می شود باید قبلا تعیین شده باشد .برای نمایش یک عدد صحیح مثبت در فرم مکمل۲، عدد مانند اعداد بدون علامت به سادگی در مبنای ۲ نوشته شده و به اندازه کافی صفر در سمت چپ آن اضافه می شود تا طول نمایش را بسازد. برای نمایش یک عدد صحیح منفی در فرم مکمل۲، ابتدا عدد را به مبنای ۲ برده، سمت چپ آن صفر اضافه می شود تا طول نمایش را بسازد سپس مکمل ۲ عدد گرفته می شود .

مثال. نمایش عدد 1116+ به فرم مکمل۲ در یک کلمه به صورت زیر نمایش داده می شود .

+1116 = 0000 0100 0101 110b =045ch

مثال: نمایش عدد 97- به فرم مکمل۲ در یک بایت

Decimal: -97
Binary:01100001b
1'Complement: 10011110
2'Complement: 10011111
= 9fh

عمل مکمل گیری مانند عمل منفی کردن است، به همین دلیل با گرفتن مکمل۲ از یک عدد منفی به عدد مثبت مطابق آن می رسیم و برعکس .

مثال. بنابراین عدد 11001000b نمایش عدد 56- در مثال اول است .

برای اعداد مثبت بیت علامت باید صفر باشد یعنی آخرین رقم هگز عدد بین ۰ تا۷ خواهد بود. عدد منفی به بیت علامت ۱ ختم می شود و می تواند ارقام بین ۸ تا f را در آخرین رقم هگز داشته باشد. وقتی علامت عدد مکمل۲ بدست آمد پیدا کردن عدد در مبنای ۲ مشکل نیست برای اعداد مثبت کافی است به مبنای ۱۰ برده شود و اعداد منفی ابتدا مکمل۲ گرفته می شود سپس به مبنای ۱۰ برده می شود .

مثال. 0d43h یک عدد مکمل ۲ با طول ۱۶ بیت است که معادل مبنای ۱۰ آن عدد 3395+ است .

مثال. b2ebh نمایش یک عدد صحیح در سیستم مکمل ۲ است معادل آن در مبنای 10 به صورت زیر بدست می آید.

b2ebh =1011001011101011b
2's Compl. =0100110100010101
=19733 ==> -19733

روش مکمل۲ اشکالات روش های قبل را برطرف می کند و تنها یک نمایش برای صفر وجود دارد و این محاسبات مکمل۲ را ساده تر از روش های قبلی می کند .جمع و تفریق نمایش مکمل ۲ همانند اعداد باینری انجام می شود. نمایش مکمل دو باعث می شود اعداد منفی یکی بیشتر از اعداد مثبت باشند(وقتی همه بیت ها 1 است).

در روش مکمل۲ اعداد قابل نمایش با طول m بیت در بازه$[-(2^{m-1}),2^{m-1}-1]$ قرار می گیرند.

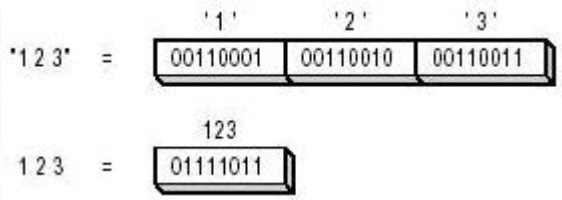
نکته ۱. تعداد بیت های عدد قبل از عمل مکمل گیری حتما باید برابر طول نمایش باشد. در غیر اینصورت عدد حاصل اشتباه است.
نکته ۲. اگر طول بیت ها برابر با طول نمایش باشد و نتوان آنرا افزایش داد حاصل اشتباه می شود.
نکته ۳. از مبنای ۱۶ برای ساده تر بیان کردن اعداد مبنای ۲ استفاده می شود .

کدگذاری ASCII

کد گذاری ASCII (American Standard Code for Information Interchange)به حروف، ارقام، علائم و کاراکترهای مختلف یک عدد باینری ۷ بیتی نسبت می دهد و هشتمین بیت را 0 درنظر می گیرد. به این صورت هر کاراکتر یک بایت را اشغال می کند .

روشن است که این روش برای نمایش اعداد مناسب نیست، چون در فرمت باینری یک بایت اعداد 0 تا 255 را نمایش می دهد، اما با کد ASCII یک بایت تنها برای نمایش یک رقم کافی است. به همین دلیل کلا این روش برای نمایش متن در حافظه استفاده می شود .

مثال. نمایش عدد 123 با دو فرمت ASCII و باینری



نوع توسعه یافته این سیستم شامل ۸ بیت برای هر کاراکتر است و ٢٥٦ حالت مختلف را شامل می شود. کدهای 0 تا 127 برای کاراکترهای استاندارد، کدهای کنترلی و ارتباطی و مقادیر 128 تا 255 برای نمایش سمبل های گرافیکی و حروف یونانی هستند .

مثال. رشته "ABC123" به صورت 41h 42h 43h 30h 31h 32h نشان داده می شود.

یک کدگذاری کامل تر که جای ASCII را دارد می گیرد Unicode است. تفاوت کلیدی بین این دو نوع کدگذاری در این است که ASCIIیک بایت را برای کدکردن یک کاراکتر استفاده می کند در حالیکه Unicode برای هر کاراکتر دو بایت را درنظر می گیرد. بنابراین کاراکترهای بیشتری را می تواند نمایش دهد که این برای نمایش کاراکترهای کلیه زبان های دنیا کاربردی است .

مثل .کدگذاری ASCII کد 41h یا ٦٥ را به کاراکتر A می دهد. کدگذاری Unicode کد 0041h هگز را می دهد .

نکته۱. تفاوت یک حرف بزرگ با یک حرف کوچک تنها در بیت شماره 5 است؛ این بیت در حروف بزرگ 0 و در حروف کوچک 1 است(01101101 ="m" و01001101 ="M").
نکته۲. ارقام 0 تا 9 کدهای 30h تا 39h را دارا می باشند.
نکته۳. کاراکترهای قابل چاپ بین 20h تا 7Eh است.
نکته٤. کاراکترهای 0 تا 1Fh و 7Fh کاراکترهای کنترلی نام دارند که قابل رویت نمی باشند.
نکته٥. کاراکتر ESC با کد 1Bh همراه با کاراکترهای دیگر اغلب برای یک عمل خاص به دستگاه های جانبی ارسال می شود .
نکته٦. کدهای 41h تا 5Ah کاراکترهای A تا Z و کدهای 61h تا 7Ah کاراکترهای a تا z هستند.
نکته۷. کاراکتر CR و LF با کدهای 0Dh و 0Ah به ترتیب باعث حرکت مکان نما به شروع خط جاری و خط بعد می شود .

کاهش و افزایش طول داده

در اسمبلی کلیه داده ها اندازه مشخص شده ای دارند. گاهی ناچار به تغییر اندازه داده هستیم. برای کاهش اندازه داده کافی است بیت های با ارزش حذف شوند. این روش برای اعداد بدون علامت و علامت دار کار می کند. قاعده کلی این است که برای اعداد بدون علامت کلیه بیت های حذف شده باید صفر باشند. و برای اعداد علامت دار بیت های حذف شده باید همگی یا 1 و یا 0 باشند. البته اگر عدد را نتوان به طرز صحیح در اندازه کوچکتر نمایش داد کاهش اندازه کار نمی کند .

افزایش داده پیچیده تر از کاهش است. عدد هگز FF را اندازه بگیرید .گسترش آن بستگی دارد که آن را چطور تفسیر کنیم. اگر آن را یک عدد بدون علامت درنظربگیریم(یعنی عدد 255)، به صورت 00FF گسترش داده می شود و اگر علامت دار باشد(یعنی ۱-) به صورت .FFFF به طور کلی برای گسترش اعداد بدون علامت کلیه بیت های جدید عدد گسترش یافته صفر می شوند ولی برای گسترش یک عدد علامت دار باید بیت علامت را بسط داد، به این معنا که.بیت های جدید بیت علامت را در خود کپی می کنند .