نویسنده : حمید رضا امیری
ایمیل: abusohrab2000@yahoo.com
نقل بخش یا تمام این درس با ذکر نام نویسنده ازاد است.

---

***بنام خدا***

1.1.2 -- سیستم شمارشی باینری :

اکثر سیستمهای مدرن کامپیوتر (از قبیل IBM PC ) با استفاده از منطق باینری عمل میکنند . کامپیوتر مقادیر را با استفاده از دو سطح ولتاژ (معمولا 0v و 5v+ ) نمایش میدهد. با یک چنین دو سطحی ما میتوانیم دقیقا دو مقدار متفاوت را نمایش دهیم . اینها میتوانند هر دو مقدار متفاوتی باشند. اما بطور قرار دادی بیایید مقادیر ۰ و ۱ را استفاده کنیم . این دومقدار تصادفا مشابه همان دو رقم بکار رفته توسط مبنای باینری است . از انجا که در اینجا یک تشابه مابین سطوح منطقی بکار رفته توسط 80x86 و دو عدد بکار رفته در مبنای باینری وجود دارد , باعث تعجب نیست که IBM PC سیستم شمارشی باینری را به خدمت گرفته است .

سیستم شمارشی باینری شبیه سیستم شمارشی دسیمال کار میکند با دو استثناء : باینری تنها ارقام ۰ و ۱ (بجای ۰ تا ۹) را میپذیرد. و باینری توان دو را بجای توان ۱۰ بکار میبرد . برای تبدیل یک عدد باینری به دسیمال , برای هر " ۱ " در رشته باینری یک 2^n را قایل شوید در جایی که n موقعیتی از شروع صفر رقم باینری است . مثلا مقدار باینری 11001010b نمایش دهنده :

2^7 + 2^6 + 2^3 + 2^1 = 128+64+8+2=202d

تبدیل دسیمال به باینری یک کمی سخت تر است . شما باید ان توانهایی از ۲ را پیدا کنید که وقتی با یکدیگر جمع میشوند , حاصل دسیمال بدست اید. ساده ترین روش اینست که از یک توان بزرگ ۲ تا پایین به سمت 2^0 کار کنید . در ذیل تبدیل مقدار دسیمال ۱۳۵۹ را ملاحظه کنید :

* 2^11 = 2048 , 2^10 = 1024 . بنابراین ۱۰۲۴ بزرگترین توان ۲ ی کوچکتر از ۱۳۵۹ است . ۱۰۲۴ را از ۱۳۵۹ تفریق کرده و مقدار باینری را با یک رقم " ۱ " از چپ اغاز میکنیم . Binary=1 , نتیجه دسیمال هست : 1359 - 1024 = 335

* توان پایینتر بعدی ۲ (2^9 = 512 ) بزرگتر ازحاصل فوق است . بنابر این یک " ۰ " به انتهای رشته باینری میافزاییم . Binary=10 نتیجه دسیمال هنوز ۳۳۵ است .

* توان پایینتر بعدی ۲ برابر ۲۵۶ (2^8 ) است .این را از ۳۳۵ تفریق کرده و یک رقم " ۱" به انتهای رقم باینری میافزاییم . Binary=101 . نتیجه دسیمال ۷۹ است.

* ۱۲۸ (2^7 ) بزرگتر از ۷۹ است . پس "۰ " دیگری به انتهای رشته باینری میافزاییم . Binary= 1010 . نتیجه دسیمال ۷۹ باقی میماند.

* توان پایینتر بعدی ۲ (2^6=64 ) کمتر از ۷۹ است . بنابر این ۶۴ را تفریق کرده و یک " ۱" به انتهای رشته باینری میافزاییم .Binary=10101 . نتیجه دسیمال ۱۵ است.

* ۱۵ کمتر از توان بعدی ۲ است (2^5=32 ) . بنابر این یک " ۰" به انتهای رشته باینری افزوده . Binary=101010 نتیجه دسیمال هنوز ۱۵ است .

* ۱۶ (2^4) بزرگتر از باقیمانده قبلی است . از اینرو یک "۰ " به انتهای رشته باینری میافزاییم . Binary=1010100 . دسیمال ۱۵ است.

* 2^3 (هشت) کمتر از ۱۵ است یک "۱" در ادامه رشته باینری قرار داده .Binary=10101001 .نتیجه دسیمال ۷ است .

* 2^2 کمتر از ۷ است . بنابر این چهار را از هفت کم کرده یک "۱" دیگر به انتهای رشته باینری میافزاییم . Binary=101010011 . نتیجه دسیمال میشود ۳ .
* 2^1 کمتراز ۳ است بنابر این یک "۱" دیگر به انتهای رشته باینری میافزاییم . Binary= 1010100111 . نتیجه دسیمال حالا ۱ میشود.
* نهایتا , نتیجه دسیمال ۱ است که همان 2^0 است . پس یک "۱" نهایی به انتهای رشته باینری می افزاییم . Binary=10101001111 .

ارقام باینری اگر چه در زبانهای سطح بالا اهمیت کمی دارند , در هر جایی از برنامه های زبان اسمبلی ظاهر میشوند.

1.1.3 -- اندازه های باینری:

در حالت محض هر عدد باینری محتوی یک تعداد بینهایت ارقام است (یا bit که مختصر و کوتاه برای ارقام باینری بکار میرود) . مثلا ما میتوانیم عدد ۵ را بصورت زیر نمایش دهیم :

101 , 00000101 , 0000000000101 , 000000000000101

هر تعداد bit صفر میتواند در ابتدای عدد باینری بیاید بدون اینکه مقدارش را تغییر دهد.نظر به اینکه 80x86 با گروههای ۸ بیتی کار میکند برایمان خیلی ساده خواهد بود که تمام اعداد باینری را تا چند ۴ بیتی یا ۸ بیتی با صفر دراز کنیم . بنابر این ما عدد ۵ را بصورت 0101b یا 00000101b نمایش میدهیم .

در امریکا مردم هر سه رقم را با یک کاما ( ویرگول ) جدا میکنند تا اعداد بزرگتر را برای خواندن ساده تر کنند . مثلا 1,023,435,208 خواناترو قابل فهم تر از 1023435208 است . ما قرار داد مشابهی را برای اعدا باینری قبول خواهیم کرد . ما هر گروه ۴ بیتی باینری را با یک فاصله خالی جدا خواهیم کرد . مثلا مقدار باینری 10101111110110010 نوشته خواهد شد:

1010 1111 1011 0010

ما اغلب چندین مقدار را با هم درون یک عدد باینری دسته بندی میکنیم . مثلا یک فرم از راهنمای 80x86 MOV کد گذاری باینری 1011 0rrr dddd dddd را برای دسته بندی سه ایتم درون ۱۶ بیت بکار میبرد : یک کد عملگر ۵ بیتی (10110 ) , یک فیلد رجیستر ۳ بیتی و یک مقدار ضروری ۸ بیتی ( dddd dddd ) . برای راحتی ما یک مقدار عددی را به موقعیت هر بیت مقرر خواهیم کرد . ما هر بیت را به شکل زیر میشماریم :

۱ ) راست ترین بیت در عدد باینری بیت سمت صفر است .

۲ ) هر بیت به سمت چپ عدد بیت متوالی بعدی تعیین میشود .

یک عدد باینری ۸ بیتی بیتهای صفر را در ۷ مکان بکار میبرد:

$X_7$ $X_6$ $X_5$ $X_4$ $X_3$ $X_2$ $X_1$ $X_0$

یک عدد باینری ۱۶ بیتی بیتهای صفر را در ۱۵ مکان بکار میبرد :

$X_{15}$ $X_{14}$ $X_{13}$ $X_{12}$ $X_{11}$ $X_{10}$ $X_9$ $X_8$ $X_7$ $X_6$ $X_5$ $X_4$ $X_3$ $X_2$ $X_1$ $X_0$

بیت صفرم معمولا به بیت low order ( L.O ) منسوب میشود . چپ ترین بیت نوعا بیت ( H.O ) high order نامیده میشود . ما بیتهای میانی را بوسیله شماره مربوطه شان منتسب خواهیم کرد.

==========

1.2 -- سازماندهی اطلاعات :

در ریاضیات محض یک مقدار میتواند یک تعداد اختیاری از بیتها را بگیرد . کامپیوتر ها از طرف دیگر با تعدادی مشخص از بیتها کار میکنند. مجموعه های عمومی و متعارف , تک بیتها , گروههای چهار بیتی ( nibbles نامیده میشوند ) , گروههای هشت بیتی ( bytes نامیده میشوند ) , گروههای ۱۶ بیتی ( words نامیده میشوند ) و الی اخر هستند . اندازه ها اختیاری نیستند . البته دلیل خوبی برای این مقادیر خاص وجود دارد . این بخش گروههای بیتی را که معمولا در چیپ های Intel 80x86 بکار رفته اند شرح خواهد داد.

### 1.2.1 -- Bits :

کوچکترین واحد داده روی یک کامپیوتر باینری یک تک بیت ( *single bit* ) است . از انجاییکه یک تک بیت قادر به نمایش دو مقدار متفاوت است ( نوعا صفر و یک ) شما ممکن است خیال کنید که اقلام خیلی کمی وجود دارند که میتوانید با یک تک بیت نمایش دهید . درست نیست ! اقلام بینهایتی وجود دارند که با یک تک بیت میتوانید نمایش دهید .

شما با یک تک بیت میتوانید هر دو ایتم مجزایی را نمایش دهید. مثلا صفر یا یک , درست یا غلط , روشن یا خاموش , مرد یازن , صحیح یا نا صحیح . بهر حال در نشان دادن انواع داده باینری ( ان اشیایی که تنها دو مقدار متمایز دارند ) بدون محدودیت هستید . شما میتوانید یک تک بیت را برای نشان دادن اعداد ۷۲۳ و ۱۲۴۵ بکار ببرید . یا شاید ۶۲۵۴ و ۵ . شما همچنین میتوانید یک تک بیت را برای نشان دادن رنگهای قرمز و ابی بکار ببرید . حتی میتوانید دو شیء یا موضوع بی ربط را با یک تک بیت نشان دهید. مثلا میتوانید رنگ قرمز و عدد ۳۲۵۶ را با یک تک بیت نشان دهید. مهم اینست که بدانید هر دو مقدار متفاوت دلخواهی را میتوانید با یک تک بیت نمایش دهید . حتی برای مغشوش تر شدن چیزها بیتهای متفاوت میتوانند نمایش دهنده چیزهای متفاوت باشند. مثلا یک بیت میتواند مورد استفاده برای نمایش مقادیرصفر و یک بکار رود , در حالیکه بیت دیگر میتواند برای نمابش مقادیردرست و غلط بکار رود. چطور میوانید با نگاه بیتها را تشخیص دهید؟ جواب اینست که البته نمیتوانید. اما این مثال تمام ایده پشت ساختمان داده کامپیو تر را روشن میکند: داده ان چیزی است که شما تعین میکنید باشد.اگر شما یک بیت را یرای نمایش یک مقدار بولی ( true/false ) بکار ببرید پس ان بیت همانطور که تعیین کردید نمایش دهنده true یا false است . برای بیتی که معنی دار میکنید باید سازگاری انرا رعایت کنید یعنی اگر یک بیت را در یک نقطه از برنامه تان برای نمایش true یا false بکار برده اید نبایستی مقدار true/false ذخیره شده در ان بیت را بعد از ان برای نمایش قرمز / ابی بکار ببرید.

از انجاییکه اکثر ایتمهایی که میخواهید مدلسازی شوند نیازمند بیش از دو مقدار متفاوت هستند , مقادیر تک بیتی عمومی ترین نوع داده ای که استفاده خواهید کرد نیستند . اما بهر حال هر چیز دیگری شامل گروههای بیتهاست . بیتها نقش مهمی در برنامه هایتان بازی خواهند کرد . البته چندین نوع داده دیگر هم وجود دارد که نیازمند دو مقدار متمایز است , بنابر این ممکن است بنظر برسد که بیتها از دبد خودشان مهم هستند و نه از دید یک نوع داده ای که میتواند دو مقدار متمایز را ذخیره کند چرا که بزودی خواهید دید که تک بیتها برای عملیات با دست خیلی مشکل هستند . بنابر این اکثر اوقات انواع دیگر داده را برای نمایش مقادیر بولی بکار خواهیم برد .

### 1.2.2 -- Nibbles :

یک nibble یک مجموعه چهار بیتی است . البته چندان یک ساختمان داده جالب توجه ویژه نیست مگر برای دو مورد : اعداد BCD ( binary coded decimal ) و اعداد هگزا دسیمال . در واقع یک تک BCD و یا یک تک رقم هگزا دسیمال چهار بیت را بر میدارد . ما میتوانیم تا ۱۶ رقم متمایز را با یک nibble نمایش دهیم ( 2^4 ) . در مورد اعداد هگزادسیمال , مقادیر 0,1,2,3,4,6,7,8,9,A,B,C,D,E,F با چهاربیت نمایش داده میشوند . BCD ده رقم مختلف را استفاده میکند ( 0,1,2,3,4,5,6,7,8,9 ) و نیازمند چهار بیت است .

در کل هر ۱۶ مقدار متمایزدلخواهی رامیتوان با یک nibble نمایش داد . اما ارقام هگزادسیمال و BCD جزو فقره های اصلی هستند که ما میتوانیم با یک تک nibble نشان دهیم .

### 1.2.3 -- Bytes :

بی هیچ سوالی , مهمترین ساختمان داده بکار رفته توسط ریز پردازنده 80x86 , بایت است . یک بایت مرکب است از هشت بیت و کوچکترین تکه داده ( *datum* ) قابل ادرس دهی روی ریز پردازنده 80x86 است . ادرسهای I/O و حافظه اصلی همگی ادرسهای بایت هستند . این معنی میدهد که کوچکترین ایتمی ( تکه ای ) که میتواند بصورت تکی و منحصر بفرد توسط یک برنامه 80x86 دست یافته شود , یک مقدار هشت بیتی است . برای دستیابی به هر چیز کوچکتر لازم است که شما بایت محتوی داده را بخوانید و بیتهای نا خواسته را پنهان ( خاموش ) کنید. بیتها در یک بایت معمولا از صفر

تا هفت شماره گذاری شده اند - شکل 1.1 - .
*بیت 0 هست low order bit یا least significant bit بایت و بیت 7 هست high order bit یا most significant bit بایت .*

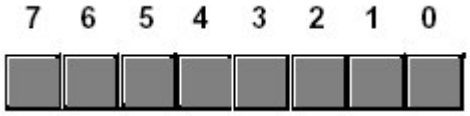


Figure 1.1: Bit Numbering in a Byte

توجه کنید که هر بایت نیزدقیقا مشتمل از دو nibble است - شکل 1.2 -.

Figure 1.2: The Twe Nibbles in a Byte

بیتهای ۰ تا ۳ شامل low oredr nibble است , بیتهای ۴ تا ۷ هم high order nibble را تشکیل میدهد . از انجاییکه یک بایت دقیقا شامل دو nibble است , پس *هر بایت برابر دو رقم هگزا دسیمال است .*

از انجاییکه یک بایت شامل هشت بیت است , میتواند 2^8 (۲۵۶) مقدار متفاوت را نشان دهد . کلا ما یک بایت را برای نمایش: مقادیر عددی در محدوده 0 تا 255 , اعداد نشان دار در محدوده 128- تا 127+ , کدهای کاراکتر ASCII / IBM , و دیگر انواع داده مخصوصی که به بیش از ۲۵۶ مقدار مختلف نیاز ندارند بکار خواهیم برد . بسیاری از انواع داده کمتراز ۲۵۶ ایتم دارند بنابراین ۸ بیت معمولا کافی است .

ازانجاییکه 80x86 یک ماشین ادرس ده بایتی است , اداره تمام یک بایت نسبت به اداره یک بیت یا نیبل تکی موثرتراست , به همین دلیل اکثر برنامه نویسان همه یک بایت را برای نشان دادن انواع داده ای که به بیش از ۲۵۶ ایتم نیاز ندارند بکار میبرند , حتی اگر کمتر از ۸ بیت برایشان کفایت کند. مثلا ما مقادیر بولی true و false را به ترتیب با $00000001_2$ و $00000000_2$ نشان خواهیم داد.

احتمالا مهمترین کاربرد یک بایت نگاه داری و جای دادن یک کد کاراکتر است. کاراکترهای تایپ شدهء در کیبورد , نمایش داده شده بر روی صفحه نمایش , و چاپ شدهء روی پرینتر,همگی مقادیر عددی دارند. برای اینکه بتوانیم با جهان گفتگو کنیمIBM PC یک متغیر مجموعه کاراکتر ASCII را بکار برد ( The ASCII Character Set ) که در ان ۱۲۸ کد برای حروف اصلی تعریف شده و ۱۲۸ مقدار ممکن باقی مانده را برای کدهای کاراکترتوسعه دادنی از قبیل کاراکترهای اروپایی, سمبلهای گرافیکی, حروف یونانی و سمبلهای ریاضی بکار برد.

### 1.2.4-- Words :

یک کلمه یک گروه ۱۶ بیتی است . ما در یک کلمه بیتها را از صفر تا ۱۵ شماره گذاری میکنیم - شکل 1.3 - .

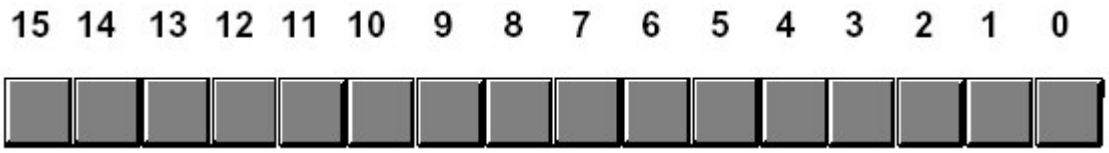


Figure 1.3: Bit Nhumbers in a Word

بیت صفر هست low order bit و بیت ۱۵ هست high order bit .
بیتهای دیگر شماره موقعیت خودشان را دارند .
بیاد داشته باشید که یک کلمه حقیقتا از دو بایت تشکیل شده است . بیتهای ۰ تا ۷ تشکیل دهنده low order bit و بیتهای ۸ تا ۱۵ تشکیل دهنده high order bit هستند - شکل 1.4-.

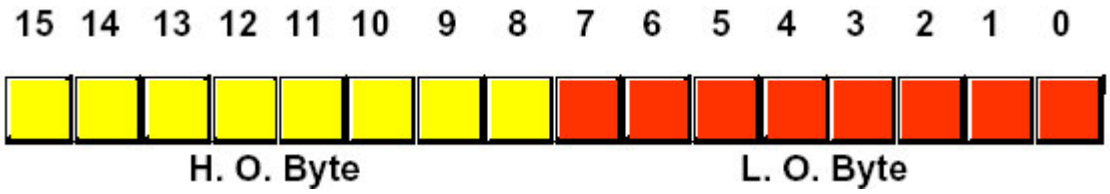


Figure 1.4: The twe Bytes in a Word

بدیهی است که یک کلمه میتواند باز هم به سمت چهار نیبل منفصل شود - شکل 1.5 -.

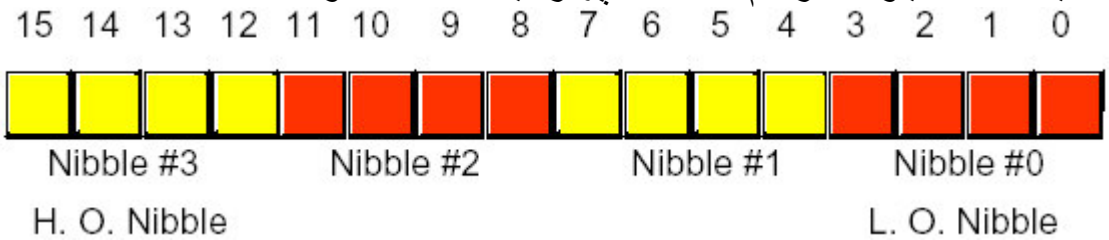


Figure 1.5 : Nibbles in a Word

نیبل صفر در کلمه low order nibble است و نیبل سه high order nibble کلمه است .دو نیبل دیگر , نیبل ۱ و نیبل ۲ هستند. با ۱۶ بیت شما میتوانید 2^16 ( ۶۵۵۳۵ ) مقدار مختلف را نمایش دهید.اینها میتوانند مقادیری در محدوده 0 تا 65535 باشند (یا 32768- تا 32767+) یا هر نوع داده دیگری با کمتر از ۶۵۵۳۶ مقدارسه مورد مصرف اصلی برای کلمات مقادیر عدد صحیح (integer) , افستها (*offsets*) و مقادیر قطعه ( *segment* ) هستند.
کلمات میتوانند مقادیر عدد صحیح را در محدوده ۰ تا ۶۵۵۳۵ یا ۳۲۷۶۸- تا ۳۲۷۵۷+ نمایش دهند.در کلمه مقادیر عددی بی نشان بوسیله مقادیر باینری متناظرو مقادیر عددی نشان دار در فرم و قالب مکمل دو نمایش داده میشوند . مقادیر قطعه , که همواره ۱۶ بیت طول دارند , تشکیل دهنده بندادرس (paragraph address) یک کد , داده , پیشوند , یا قطعه پشته در حافظه هستند.

### 1.2.5 -- Double Words :

یک دوکلمه ای دقیقا همان چیزی است که نامش دلالت میکند , یک زوج کلمه . بنابراین کمیت و اندازه یک دو کلمه ای به طول ۳۲ بیت است - شکل 1.6 -.

Figure 1.6: Bit Numbers in a Double Word

طبیعتا این دو کلمه ای به یک low order word و یک high order word یا چهار بایت مختلف یا هشت نیبل مختلف تقسیم شده - شکل 1.7 -.

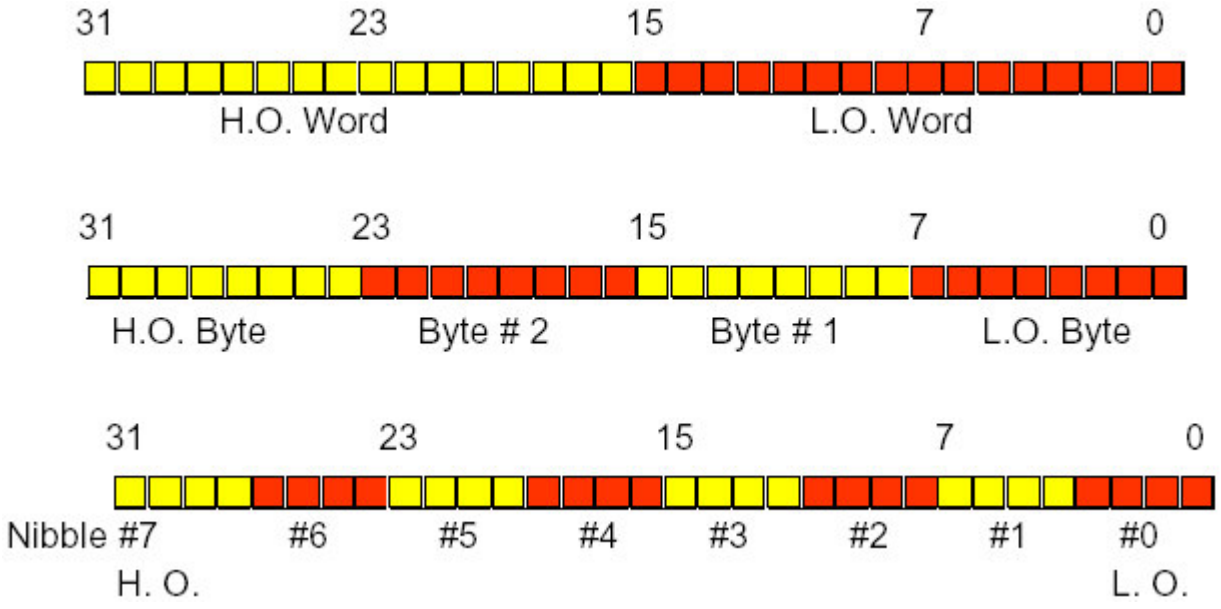


Figure 1.7: Nibbles,Bytes,and Words in a Double Word

دو کلمه ای ها تمام انواع چیزهای متفاوت را میتوانند نمایش دهند . اولین و پیشترین در فهرست, نمایش یک ادرس قطعه بندی شده است .اقلام معمول دیگر نشان داده شده با یک دو کلمه ای یک مقدار عدد صحیح ۳۲ بیتی (منظور اعداد بی نشان در محدوده 0 تا 4,294,967,295 یا اعداد نشان دار در محدوده 2,147,483,648- تا 2,147,483,647+) است .مقادیرممیزشناور ۳۲ بیتی نیز در یک دو کلمه ای میگنجد. دراکثرمواقع ما دوکلمه ای ها را برای نگهداری ادرسهای قطعه نبدی شده بکار میبریم.