

x86 Assembly Language FAQ – General Part II

Source: <http://coding.derkeiler.com/Archive/Assembler/comp.lang.asm.x86/2004-06/0029.html>

From: fys (zfysz_at_cybertrails.com)

Date: 06/03/04

Date: Wed, 2 Jun 2004 22:44:09 +0000 (UTC)

x86 Assembly Language FAQ – General Part II

From: fys_at_cybertrails.com (Ben Lunt)

Subject: x86 Assembly Language FAQ – General Part 2 of 3

Supersedes: <[89sf4a\\$stn\\$6@news.dgsys.com](mailto:89sf4astn6@news.dgsys.com)>

Followup-To: alt.lang.asm,comp.lang.asm.x86

Date: 3 Mar 2004 23:04:44 MST

Organization: Forever Young Software

Lines: 1212

Approved: news-answers-request@MIT.EDU

Distribution: world

Expires: Tue, 22 Jun 2004 23:59:59 GMT

Message-ID: <[8b8v6c\\$n7i\\$6@news.dgsys.com](mailto:8b8v6c$n7i$6@news.dgsys.com)>

Reply-To: fys@cybertrails.com

Summary: This is the FAQ for the x86 Assembly Language programmers for the alt.lang.asm and comp.lang.asm.x86 newsgroups. This particular section of the FAQ is part two of three parts that contain x86 assembly language information common to all assemblers.

Keywords: x86 Assembly Language ASM FAQ General

Archive-Name: assembly-language/x86/general/part2

Posting-Frequency: monthly (21st of every month)

Last-modified: 2004/03/03

Subject: 15. Accessing 4 Gigs of Memory in Real Mode

Flat real mode is a popular name for a technique used to access up to 4 GB of memory, while remaining in real mode. This technique requires a 80386 or higher processor. The address space really is not flat.

Actually, this technique allows you treat one or more segments as large (32-bit) segments, thereby accessing memory above 1 MB.

When the CPU accesses memory, the base address of the segment used is not described by the value currently in the appropriate register. The value is stored internally in a structure known as the descriptor cache. Changing the value of a segment register results in that segment's entry in the descriptor cache being recalculated according to the rules of the current mode. In real mode, the value of the segment register is shifted left four bits to find the base address of the segment, and the

size of the segment is always 64k. In protected mode, the value in the segment register is used as an index into a descriptor table located in memory, and the base address and size (which may be as small as 4 KB, or as large as 4 GB) from the descriptor table are loaded into the descriptor cache.

When the processor changes modes, the contents of the processor's internal descriptor cache are not changed. The reason is because changing them would result in (at the very least) the code segment being recalculated according to the new mode's rules, most likely causing your program to crash. Thus the program must load the segment registers with sensible values after the mode switch occurs. Consider an example where real mode code is located in segment 1000h. If switching modes caused an immediate recalculation of the descriptor cache, the processor would attempt to read entry 1000h of the descriptor table immediately upon switching to protected mode. Even if this were a valid descriptor (unlikely), it would have to have a base address identical to real mode segment 1000h (i.e., 10000h), and a size limit of 64 KB to prevent a probable crash. An invalid descriptor would cause an immediate processor exception.

Normally, aside from preventing situations like that in the above example, there is little to be said about this feature. After all, as soon as you reload new values into the segment register, the descriptor cache entry for that segment will be reset according to the rules of the current mode. After switching from protected mode to real mode, however, when you load the segment registers with their new values, the segment's base address is recalculated according to real mode rules, but the size limit is not changed. After setting the 4 GB limit (which must be done in protected mode), it will stay in place until changed by another protected mode program, regardless of what values are loaded in the segment register in real mode.

So, the steps to using this technique are as follows:

1. Set up a bare bones global descriptor table, with a null entry, and a single entry for a 4 GB segment. The base address of this segment is not important.
2. If you don't wish to define an interrupt descriptor table (IDT), you must disable interrupts before switching to protected mode. You do not need a full-fledged protected mode environment for this, so it is easiest just to disable interrupts and not worry about the IDT.
3. Switch to protected mode.
4. Load the segment registers you wish to change with the selector for the 4 GB segment. I recommend using FS and/or GS for this purpose, for reasons I'll describe below.
5. Return to real mode.
6. Re-enable interrupts.

After these steps, you can then load your segment registers with any value you wish. Keep in mind that the base address will be calculated according to real mode rules. Loading a value of 0 into a segment

register will result in a 4 GB segment beginning at physical address 0. You can use any of the usual 32-bit registers to generate offsets into this segment.

Some points to keep in mind:

1. Some software depends on 64 KB segment wrap-around. While rare, it is possible that you will encounter software that crashes if the older segments (DS or ES) are 4 GB in size. For that reason, I recommend only using FS and/or GS for this purpose, as they are not used as widely as the others.

2. You should never change the limit of the code segment. The processor uses IP (not EIP) to generate offsets into the code segment in real mode; any code beyond the 64 KB mark would be inaccessible, regardless of the segment size.

3. You should never change the limit of the stack segment. This is similar to the above; the processor uses SP in real mode, rather than ESP.

4. Because of the necessity of switching to protected mode, this technique will not work in a virtual 8086 mode "DOS box" from Windows, OS/2, or any other protected mode environment. It only works when you start from plain, real mode DOS. Many memory managers also run DOS in V86 mode, and prevent the switch to protected mode. It is possible to use VCPI to work around this, but if you go to that length you will probably find that you have implemented a complete protected mode environment, and would not need to return to real mode anyway.

5. This technique will not work in the presence of any protected mode software that changes segment size limits. When that software returns control to your real mode program, the limits will be the values to which the protected mode code set them. If these limits are different that what your program used, problems can result. At the very least, your program will return incorrect results when accessing data stored in extended memory. At worst, your program will crash and burn.

The benefits of this technique are many. Most importantly, you can access extended memory without resorting to slow BIOS calls or having to implement a complete DOS extender. If your program uses interrupts extensively (timer interrupts for animation or sound, for example), real mode is a better choice because protected mode handles interrupts slower. DOS itself uses this technique in HIMEM.SYS as a fast, practical method of providing access to extended memory.

Code demonstrating this technique is available:

<http://sunsite.lanet.lv/ftp/mirror/x2ftp/msdos/programming/memory/realmem.zip>

For further reading on this topic, I suggest "DOS Internals," by Geoff Chappell. It is published by Addison-Wesley as part of the Andrew Schulman Programming Series. The ISBN number is 0-201-60835-9.

Contributor: Sherm Pendley, grinch@access.mountain.net
Last changed: 03 Mar 2004

Return to the Table Of Contents-----

Subject: 16. What Is Available at developer.intel.com

16.1 PENTIUM & PENTIUM PRO INFORMATION

The gateway for information on the Pentium family of processors at Intel are:

<http://developer.intel.com/design/pentium>

<http://developer.intel.com/design/pro>

<http://developer.intel.com/design/pentiumiii/>

Information linked to this page is: Application Notes, Datasheets, Manuals, Specification Updates, and much more.

16.2 INTEL DEVELOPMENT TOOLS

The below page has links to software, hardware, evaluation kits and documentation on Intel OEM products. Areas covered are Intel Software Performance Products, Internet Technologies, Multimedia and Intel Products.

<http://developer.intel.com/design/develop.htm>

16.3 INTEL TECHNOLOGIES

Intel has overviews, in-depth system architecture tutorials and specifications on a variety of PC platform and communications technologies. Areas covered are MMX Technology, Intelligent I/O, WinSock 2, and much more.

<http://developer.intel.com/design/tech.htm>

16.4 GET INTEL'S WEB SITE ON CDROM

You can no longer order the following CDROM's.

Have you been spending a long time on line downloading one of the many manuals available from Intel's Developer Web Site. Now you can get the entire Technology and Product portions of that web site available on CDROM. You access the CDROMs with your browser. It now takes longer to launch the Acrobat reader than to download a meg .pdf file. With the Aug 98 version, the package includes three CD-ROMs: Products and Product Selectors; Tools and Motherboards; and Technologies.

Unfortunately Intel has stopped this service. See

<http://developer.intel.com/design/litcentr/cdorder.htm>
for a reason why.

However, it seems that you can still get the last version issued.

The last version was May 1999. It appears that you can still order it at:

http://apps.intel.com/scripts-order/viewbasket.asp?SKURev=273000_011&site=developer&LNavFile=TRUE

16.5 Intel 80386 Programmer's Reference Manual

This is a very popular Intel Manual that is no longer available for downloading from Intel. Luigi Sgro has translated it into HTML and is available:

<http://www.baldwin.cx/386htm/toc.htm>

Contributor: Raymond Moon, raymoon@moonware.dgsys.com
Last changed: 03 Mar 2004

Return to the Table Of Contents-----

Subject: 17. Interrupts and Exceptions

"(with interrupts) the processor doesn't waste its time looking for work – when there is something to be done, the work comes looking for the processor."

– Peter Norton

INTERRUPTS AND EXCEPTIONS

Interrupts and exceptions both alter the program flow. The difference between the two is that interrupts are used to handle external events (serial ports, keyboard) and exceptions are used to handle instruction faults, (division by zero, undefined opcode).

Interrupts are handled by the processor after finishing the current instruction. If it finds a signal on its interrupt pin, it will look up the address of the interrupt handler in the interrupt table and pass that routine control. After returning from the interrupt handler routine, it will resume program execution at the instruction after the interrupted instruction.

Exceptions on the other hand are divided into three kinds. These are Faults, Traps and Aborts. Faults are detected and serviced by the processor before the faulting instructions. Traps are serviced after the instruction causing the trap. User defined interrupts go into this category and can be said to be traps; this includes the MS-DOS INT 21h software interrupt, for example. Aborts are used only to signal severe system problems, when operation is no longer possible.

See the below table for information on interrupt assignments in the Intel 386, 486 SX/DX processors, and the Pentium processor. Type specifies the type of exception.

Vector number Description

0 Divide Error (Division by zero)

1 Debug Interrupt (Single step)

- 2 NMI Interrupt
 - 3 Breakpoint
 - 4 Interrupt on overflow
 - 5 BOUND range exceeded
 - 6 Invalid Opcode
 - 7 Device not available (1)
 - 8 Double fault
 - 9 Not used in DX models and Pentium (2)
 - 10 Invalid TSS
 - 11 Segment not present
 - 12 Stack exception
 - 13 General protection fault
 - 14 Page fault
 - 15 Reserved
 - 16 Floating point exception (3)
 - 17 Alignment check (4)
 - 18 – 31 Reserved on 3/486, See (5) for Pentium
 - 32 – 255 Maskable, user defined interrupts
-

- (1) Exception 7 is used to signal that a floating point processor is not present in the SX model. Exception 7 is used for programs and OSs that have floating point emulation. In addition, the DX chips can be set to trap floating point instructions by setting bit 2 of CR0.
 - (2) Exception 9 is Reserved in the DX models and the Pentium, and is only used in the 3/486 SX models to signal Coprocessor segment overrun. This will cause an Abort type exception on the SX.
 - (3) In the SX models this exception is called 'Coprocessor error'.
 - (4) Alignment check is only defined in 486 and Pentiums. Reserved on any other Intel processor.
 - (5) For Pentiums Exception 18 is used to signal what is called an 'Machine check exception'.
- The other interrupts, (32–255) are user defined. They differ in use from one OS to another.

For a list of MS–DOS interrupts, see 'Obtaining HELPPC' (Subject #6) or Ralf Browns Interrupt List (Subject #11)

Contributor: Patrik Ohman, patrik@astrakan.hgs.se
Last changed: 03 Mar 2004

Return to the Table Of Contents-----
Subject: 18. ASM Books Available
The format is Author, Title, Level, and short description

Ray Duncan
Advanced MSDOS Programming
Advanced
Both a tutorial and a reference for MS–DOS capabilities and services, including reference sections on DOS function calls, IBM ROM BIOS, mouse driver and LAM. expanded memory. Excellent quality example programs

throughout.

By Peter Norton and John Socha

Peter Norton's Assembly Language Book For the IBM PC

Novice

Good for an introduction to Assembly Language. Plenty of programming examples. Older versions of this book used to have a sample disk. As you read the book, you slowly add on code to what eventually is Disk Patch – the book's version of Norton's commercially available Disk Edit program. Great for complete beginners seeking novice rank.

Maljugin, Izrailevich, Sopin, and Lavin

The Revolutionary Guide to Assembly Language

Novice

This is one of the best introductory texts I have ever seen. There are so many authors that the topic is broken down into specific categories: video, BIOS, keyboard, etc.. Most intro texts force you to follow a set plan of learning assembly, but in this book you can turn to a specific topic almost immediately. It is so-so as a reference book, however – a few tables of interrupts in the back.

Maljugin, Izrailevich, Sopin, and Lavin

Master Class Assembly Language

Advanced

Review: This is the sequel to The Revolutionary Guide To Assembly Language. Equally thick and massive, it covers many of the topics we see today – hardware interfaces, sound cards, data compression, even protected mode programming. Brief review of assembly at the beginning, but moves very quickly. Read this if you are intermediate seeking expert status. Definitely not recommended for beginners. If you are a beginner and you think you like the topics covered in this book, buy the one before it too. Also comes with a disk of source code examples from the book (MASM highly recommended, not TASM).

Alan Wyatt

Advanced Assembly Language

Advanced

This book's best feature is its comprehensive guide on device drivers. There are good chapters on controlling the mouse, file access, using memory, etc.

Ralf Brown and Jim Kyle

PC Interrupts – 2nd Edition

Intermediate/Advanced

The definitive book on interrupt programming for PCs and compatibles. Based on the freeware Interrupt List by Ralf Brown

For an extensive book list without descriptions, point your web browser to:

<http://www.alaska.net/~rrose/book.htm> [Broken]

Sites with more books (some with reviews) are:

<http://www.mavrahane.com/program/assembly/assembly/ass.htm#Books> [Broken]

<http://www.cet.com/~jvahn/80xbook.html> (short descriptions)

<http://www.cybertrails.com/~fys/books.htm>

(ASM books as well as General DOS programming books)

Contributors: Antonio Alonso, Solomon Chang, Paul Gilbert, Dave Navarro, Mike Schmit and James Vahn.

Last changed: 03 Mar 2004

Return to the Table Of Contents-----

Subject: 19. ASM Code Available on the Internet

19.1 SIMTEL SITES

The SimTel has a directory devoted to assembly language.

<ftp://ftp.simtel.net/pub/simtelnet/msdos/asmurl>

or

<http://www.simtel.net/category.php?id=16&SiteID=simtel.net>

19.2 80xxx Snippets

Fidonet's echo for 80xxx programming has a collection of code that is maintained by Jim Vahn, jvahn@short.circuit.com. The collection is on the web. In addition to downloading the snippets there is an assembly language related book list. The URL is:

<http://www.cet.com/~jvahn>

The ability to get these files via e-mail has been discontinued.

19.3 X2FTP.OULU.FI

This ftp site, [x2ftp oulu.fi](ftp://x2ftp oulu.fi), has some ASM source code not available at the SIMTEL sites. The following describes some directories and the type of information that is available in them.

This server seems to be down, so I have included links using one of its mirrors.

Protected mode utilities and some source code:

<ftp://x2ftp oulu.fi/pub/msdos/programming/pmode>

[Mirrored at]

<http://sunsite.lanet.lv/ftp/mirror/x2ftp/msdos/programming/pmode>

Some asm code:

<ftp://x2ftp oulu.fi/pub/msdos/programming/source>

[Mirrored at]

<http://sunsite.lanet.lv/ftp/mirror/x2ftp/msdos/programming/source>

<ftp://x2ftp.oulu.fi/pub/msdos/programming/progrsrc>
[Mirrored at]
<http://sunsite.lanet.lv/ftp/mirror/x2ftp/msdos/programming/progrsrc>

19.4 JUMBO

It seems that JUMBO has dropped their DOS support.

19.5 THEREEF

I just found another site that carries this asm source code. This site has source code and information that I have not found elsewhere.

<http://marina.mfarris.com/theref/theref.html>

19.6 PC GAMES PROGRAMMER ENCYCLOPEDIA

This encyclopedia is a collection of files related to game programming. Many of these files contain programming examples. Topics included are ASM tutorial, VGA and SVGA programming information, graphic algorithms, graphic file formats, soundcard and other PC hardware programming information. This encyclopedia is available online at the PC-GPE web page:

<http://www.qzx.com/pc-gpe/>
[Re-routed to:]
<http://brand107.home.comcast.net/pc-gpe/>

19.7 PROGRAMMERS DISTRIBUTION NETWORK ASSEMBLY LANGUAGE FILES

These files appear to be a mirror of the assembly-related files distributed on FidoNet by PDN. There is one that is a must if you want to write ASM WinNT and Win95 applications. It is walk32_1.zip. Walk32 is a complete app and dll development kit with linker and includes files, libraries, tools, and many samples. MASM 6.x required.

<http://www.programmersheaven.com/search/Download.asp?FileID=355>

The page to all of Programmer's Heaven ASM files is:

<http://www.programmersheaven.com/zone5/index.htm>

19.8 TENIE REMMEL'S ASSEMBLY SNIPPETS CODE COLLECTION

The Assembly Snippets is a large collection of assembly language code and other information. Many files from the original 80XXX snippets, the ASM0-Z collection, and the Aquila site are included. All code is 99% guaranteed to compile under TASM. This new release contains the following items, among others:

- An object file disassembler
- A 4971 byte Tetris game
- Several Conway LIFE programs
- Assembly & Disassembly tables

A demonstration of FakeMode
Several powerful editors
A complete DOS extender
A Pentium optimization list
A ModeX graphics library
Info for writing antivirus

You can download these rather large files from Programmer's Heaven:

<http://www.programmersheaven.com/zone5/index.htm>

Contributor: Raymond Moon, raymoon@moonware.dgsys.com
Last changed: 03 Mar 2004

Return to the Table Of Contents-----

Subject: 20. How to Commit A File

The easiest solution is to open or create the file to be committed using Int 21h function 6ch, extended open/create. The BX register contains the desired Open Mode. One option that can be or'ed into this register is what Microsoft calls, OPEN_FLAGS_COMMIT, that has the value of 4000h. Using this option caused DOS to commit the file after each write. This function has been available (documented) since DOS 4.0.

If you do not want to commit the file at each write but only when certain conditions are met, use Int 21h function 68h, commit file. This function has been available (documented) since DOS 3.3.

If you need to support versions of DOS before 3.3, the following technique will flush the all stored data without closing and opening the file. The time consuming process is the opening of the file.

1. Use 21h function 45h to create a duplicate file handle to the file to be flushed.
2. Close that duplicate file handle.

This technique will work all the way back to DOS 2.0.

Contributor: Raymond Moon, raymoon@moonware.dgsys.com
Last changed: 03 Mar 2004

Return to the Table Of Contents-----

Subject: 21. Using Extended Memory Manager

21.1 HOW TO USE XMS

XMS usage – short recipe:

1. Verify have at least 286 (pushf; pop AX; test AX,AX; js error).
2. Verify vector 2Fh set (DOS 3+ sets it during boot).
3. AX=4300h, Int 2Fh, verify AL=80h (means XMS installed).
4. AX=4310h, Int 2Fh, save ES:BX as dword XmsDriverAddr.
5. AH=8, call [XmsDriverAddr] – returns ax=largest free XMS memory block size in kB (0 if error).
6. AH=9, DX=required size in kB, call [XmsDriverAddr] – allocates memory (returns handle in DX – save it).
7. AH=0Bh, DS:SI->structure {
 dword size (in bytes and must be even),

```
word source_handle,  
dword source_offset,  
word destination_handle,  
dword destination_offset }
```

(if any handle is 0, the "offset" is Real Mode segment:offset)

8. AH=0Fh, BX=new size in kB, DX=handle, call [XmsDriverAddr] – changes memory block size (without losing previous data).
9. AH=0Ah, DX=handle, call [XmsDriverAddr] – free handle and memory.

Initially, should process #1–#6, then can use #7 to put data in/get data from XMS memory, or #8 to change XMS memory block size. On exit, use #9 to free allocated memory and handle.

Hint: handle cannot be 0, since zero is used as "no handle allocated" value.

Errors for XMS calls (except AH=7 – Query A20) are signaled by AX=0.

Error code returned in BL, few codes can check for are:

- 80h – not implemented,
- 81h – VDISK detected (and it leaves no memory for XMS),
- 82h – A20 error (e.g., fail to enable address line A20),
- A0h – all allocated,
- A1h – all handles used,
- A2h – invalid handle,
- A3h/A4h – bad source handle/offset,
- A5h/A6h – bad destination handle/offset,
- A7h – bad length,
- A8h – overlap (of source and destination areas on copy),
- A9h – parity error (hardware error in memory),
- Abh – block is locked,
- 00h – OK

For more info read INT 2Fh, AH=43h in Ralf Brown interrupt list.

21.2 WHAT IS THE 'LINEAR BLOCK ADDRESS' RETURNED BY LOCK MEM BLOCK?

When you lock mem block, XMS driver arranges memory governed by it in a way the locked block forms one contiguous area in linear address space and returns you starting address of the memory. Linear address is base address of segment + offset in segment, in Real Mode it is $\text{segment} * 16 + \text{offset}$, in Protected Mode the base address is kept in LDT or GDT; note offset can be 32-bit on 386+. If paging is not enabled, linear address = physical address. You do not need the linear address unless you use 32-bit offsets in Real Mode or you use Protected Mode (see previous answer for explanation of how you can access XMS memory).

Contributor: Jerzy Tarasiuk, JT@zfja-gate.fuw.edu.pl
Last changed: 03 Mar 2004

Return to the Table Of Contents-----
Subject: 22. EXE2BIN Replacement
A utility, EXE2BIN, used to be included in DOS. This utility was needed

to convert the output of the linker from .EXE to .COM format because the linkers could not do this directly. As linkers became more capable, the need for this utility vanished, so EXE2BIN was dropped from DOS. If you still are using an older assembler and linker, you now have been left out in the cold. Well, not quite, as there are three shareware equivalent programs.

22.1 EXECOM14.ZIP

EXECOM was written by Chris Dunford in C. The .zip file contains the executable, documentation and the .c source that Chris Dunford has released into the public domain. The current version is 1.04 with a 2 Mar 88 date.

<ftp://ftp.simtel.net/pub/simtelnet/msdos/asmutl/execom14.zip>

22.2 BIN.ZIP

This replacement version was written by Bob Tevithick. It is based upon versions 1.00 of Chris Dunford's program. The .zip file contains only the executable and documentation. No source is included.

<ftp://ftp.simtel.net/pub/simtelnet/msdos/asmutl/bin.zip>

22.3 X2B11.ZIP

X2B is written in 100% assembly language by Henry Nettles. Again, it is based upon Chris Dunford's program. The zip file contains the executable and .asm source. The documentation is in the source code.

<ftp://ftp.simtel.net/pub/simtelnet/msdos/asmutl/x2b11.zip>

22.4 EXE2COM.ZIP

<http://www.cybertrails.com/~fys/zips/exe2com.zip>

22.5 THE REAL THING, EXE2BIN.EXE

If you need the real thing, EXE2BIN.EXE is available on the DOS Supplemental Diskettes. These disks can be downloaded from Microsoft.

for MS DOS 6.0

<ftp://ftp.microsoft.com/peropsys/msdos/public/supplmnt/DOS6SUPP.EXE>

for MS DOS 6.2

<ftp://ftp.microsoft.com/peropsys/msdos/public/supplmnt/DOS62SP.EXE>

for MS DOS 6.21

<ftp://ftp.microsoft.com/peropsys/msdos/public/supplmnt/SUP621.EXE>

for MS DOS 6.22

<ftp://ftp.microsoft.com/peropsys/msdos/public/supplmnt/SUP622.EXE>

Contributor: Raymond Moon, raymoon@moonware.dgsys.com

Last changed: 03 Mar 2004

Return to the Table Of Contents-----

Subject: 23. ASM Tutorials Available on the Internet

There are several assembly language tutorials available on the Internet.

23.1 FROM SIMTEL MIRRORS

>>From the SimTel Mirrors, e.g., oak.oakland.edu, there are two tutorials available in the simtel/msdos/asmutil directory.

<ftp://ftp.simtel.net/pub/simtelnet/msdos/asmutl/asmtutor.zip>

The tutorial is by Joshua Averbach. It is old, dated Jun 1988, and designed for the 8088 processor.

<ftp://ftp.simtel.net/pub/simtelnet/msdos/asmutl/primer2.zip>

This tutorial is designed specifically for the cheap assembler (CHASM) also available in this directory.

23.2 GAVIN ESTEY'S TUTORIAL

A tutorial has been written by Gavin Estey. He has provided his tutorial in HTML format available at:

<http://burks.brighton.ac.uk/burks/language/asm/>

23.3 VLA'S ASSEMBLY LANGUAGE TUTORIAL

This tutorial is available directly or as part of the PC Games Encyclopedia:

<http://sunsite.lanet.lv/ftp/mirror/x2ftp/msdos/programming/gpe/pcgpe10.zip>

or on-line at:

<http://brand107.home.comcast.net/pc-gpe/>

23.4 ASM Tutorial on University of Guadalajara Web Site

The on-line tutorial is available:

<http://thsun1.jinr.ru/~alvladim/man/asm.html>

23.5 RANDALL HYDE'S ART OF ASSEMBLY LANGUAGE

Randy Hyde's Assembly Language Course Material. This in my opinion is the best assembly language tutorial available on the Internet.

<http://webster.cs.ucr.edu/AoA/index.html>

Do not miss his Assembly Language Style Guide.

<http://webster.cs.ucr.edu/Articles/asmstyle.pdf> .pdf version

<http://webster.cs.ucr.edu/Articles/asmstyle.html> HTML version

23.6 PATRICK STUDDARD'S ASSEMBLY CLASS NOTES

Patrick Studdard has a very extensive library of supplementary class notes for assembly language. These are available for all and not just those who are taking the class. They are available:

<http://www.csis.american.edu/~studdard/classes/fall1995/4028201/notes/index.html>

[Broken]

23.7 TORE NILSSON'S ASSEMBLY TUTORIAL PAGE

VLA's Assembly and DMA programming tutorials, Asphyxia's VGA tutorials, and some graphics and sound programming information.

<http://www.ice-digga.com/programming/index2.html>

23.8 HOMER TILTON'S ASSEMBLY LANGUAGE TUTORIAL

ZDNet offers an Assembly Language tutorial by Homer Tilton. To find it, use the following URL:

<http://www6.zdnet.com/cgi-bin/taxis/swlib/hotfiles/info.html?fcode=000804>

[broken]

23.9 Mike Babcock's ASM Tutorial

Mike Babcock has a small tutorial. Unfortunately, all the links on the page currently are broken. The basic URL is:

<http://w3.tyenet.com/mbabcock/prg.asmtut1.html>

(Note that the internal links currently are broken. I have contacted the author, and he has replied that he will be correcting this shortly.)

23.10 BRIAN BROWN'S CENTRAL INSTITUTE OF TECHNOLOGY COURSE WARE

Brian Brown has a very good tutorial along with others. The assembly language tutorial, version 3.1, starts:

<http://goforit.unk.edu/asm/default.htm>

23.11 FERDI SMIT ASSEMBLY LANGUAGE TUTORIAL

Ferdi Smit has a nice tutorial in text and HTML. It is available:

<http://www.xs4all.nl/~smit/docs.htm#asm>

23.12 PROF. LOCKWOOD'S EE291 CLASS LECTURE NOTES

Prof. Lockwood's class lecture notes, resources, etc. are a very good source of information on assembly language programming. His URL is:

<http://www.ece.uiuc.edu/~ece291/>

Contributor: Raymond Moon, raymoon@moonware.dgsys.com

Last changed: 03 Mar 2004

Return to the Table Of Contents-----

Subject: 24. Shareware and Freeware Assemblers

24.1 A86 v4.05

Highest Processor: 80286 (A86)

Highest Processor: Intel Pentium Family (A386 only)

Limitations: none

Supported: by the author <http://ejl.com/>

Platform: DOS

Output: DOS (.COM/.OBJ)

Type: Freeware

Comments: Slightly different syntax, but good assembler

For more details, see the A86 section of this FAQ.

<http://www.cybertrails.com/~fys/faq/a86.html#3>

To get the A86/D86 now:

<http://ejl.com/a86.zip>

24.2 CHASM/CHASM4.ZIP v4.00

Highest Processor: 8088 (8086)

Limitations: none

Supported: unknown

Platform: DOS

Output: DOS (.COM/BLOAD/TP)

Type: Annoyware/Crippleware

Comments: outdated but a good learning tool

This assembler was the first shareware assembler available. CHASM was written Mr. David Whitman. The current version available is version 4 and dated in 1983. This version supports only the 8088 (8086) processor, and the output only is:

.COM file (.EXE is not supported)

BLOADable – format for interpreted BASIC to load and execute

External procedure for TurboPascal – TurboPascal version not given

The version available on the internet is annoyware and crippleware. For \$40 registration fee, you will get the complete version without the annoying banner page. This version supports macros, conditional

assembly, include files, operand expressions and structures.

I do not recommend this assembler because of its limited capability and it is very out of date. Its URL is:

<ftp://ftp.simtel.net/pub/simtelnet/msdos/asmutl/chasm4.zip> (crippleware)

24.4 THE ARROWSOFT ASSEMBLER, VALARROW.ZIP v1.00d

Highest Processor: 80286

Limitations: source file to 64k

Supported: Rick Elbers (see below)

Platform: DOS

Output: DOS (.COM/???)

Type: shareware

Comments: outdated but a good learning tool

This assembler is the public domain version of the Professional Arrowsoft Assembler by Arrowsoft Systems, Inc. The version is 1.00d and is dated in 1986. This assembler is a MASM 3.0 compatible assembler and supports up to 80286 processor. Compared to the Professional version, the public domain version has one major limitation. The source file size is limited to 64K bytes.

The file also includes a public domain linker, full screen editor and an EXE2BIN clone program.

<ftp://ftp.simtel.net/pub/simtelnet/msdos/asmutl/valarrow.zip>

Rick Elbers maintains several web pages dedicated to this assembler. If you use this assembler, visit this site.

<http://www.geocities.com/SiliconValley/Heights/7052/valarr.html>

24.5 WOLFWARE ASSEMBLER, WASM223.ZIP v2.23

Highest Processor: 80286

Limitations: unknown

Supported: unknown

Platform: DOS

Output: DOS (.COM/.OBJ)

Type: freeware

Comments: outdated but a good learning tool

This assembler was written by Mr. Eric Tauck. The latest version is 2.23 and dates from 1991. This assembler supports up to the 80286 processor. It will assemble directly into a .COM file or .OBJ file. It supports a simplified syntax and program structure so programs written for this assembler may not be compatible with other assemblers. Several source files for programs are included with the .zip file.

It is available from the author at:

<http://www.catherders.com/dirs/PCB/ASM/WASM223.ZIP>

24.6 MAGIC ASSEMBLER, ASM112.ZIP v1.12

Highest Processor: unknown

Limitations: unknown

Supported: unknown

Platform: DOS

Output: DOS (.COM/.BIN)

Type: freeware

Comments:

The version is 1.12 and dates from Oct 2000. This assembler was written by Mr. Bert Greevenbosch. The output is either a .COM file or a boot sector program. The assembly commands are standard except for the jump and call commands. Again, the source code will not be compatible with other assemblers. Beware of version 1.04. That version had a bug that when executed without the print command, the assembler terminated with a runtime error. This is corrected in subsequent versions.

<ftp://ftp.simtel.net/pub/simtelnet/msdos/asmutl/asm112.zip>

24.7 GEMA, GEMA.ZIP v2.6a

Highest Processor: Pentium Family

Limitations: unknown

Supported: unknown

Platform: DOS

Output: DOS (.COM/.EXE)

Type: unknown

Comments:

This assembler revision is 2.6a with a date, 7 Jan 96. It is different from all other x86 assemblers I have seen. This assembler is based upon Motorola's 68k mnemonics and logical structure. All instructions, Pentium Pro and known undocumented are supported. GEMA was designed especially for 32-bit processing. The assembler will take only one source code file and will output a .COM or .EXE file. No linker is required. DESA.EXE, a beta GEMA disassembler is available in the GEMA package. ASM2GEMA.EXE, a TASM to GEMA translator is no longer available as part of the GEMA package. An interactive real and protected-mode debugger is in progress.

This assembler is available from:

<http://www.programmersheaven.com/zone5/cat25/1346.htm>

24.8 NASM v0.98.36

Highest Processor: Pentium Family

Limitations: none

Supported: <http://nasm.dhs.org/> [broken]
Platform: DOS/Linux?
Output: DOS/Linux (.out/.ELF/.COFF/.OBJ/Win32)
Type: freeware
Comments: Slightly different syntax, but many output platforms

The birth of this assembler started out of a thread that started on comp.lang.asm.x86. When you download this assembler, you get the source code in ANSI C. The web page devoted to this assembler is:

<http://sourceforge.net/projects/nasm>
<http://nasm.dhs.org/> [broken]

NASM is an 80x86 assembler designed for portability and modularity. It supports a range of object file formats including Linux a.out and ELF, COFF, Microsoft 16-bit OBJ and Win32. It will also output plain binary files. Its syntax is designed to be simple and easy to understand, similar to Intel's but less complex. It supports Pentium, P6 and MMX opcodes, and has macro capability. It includes a disassembler as well.

Major new features present in this release include:

1. The long-awaited listing file support!
2. Support for a search path for include files.
3. OS/2 object file support, although it's experimental as yet (could anyone with OS/2 _please_ give it a testing for me?).
4. This release, and all NASM releases from now on, include pre-built Win32 versions of NASM and NDISASM, as well as the 16-bit DOS versions.
5. Numerous bug fixes, including the repeatedly-reported bug about blank lines in macro definitions, and the one that prevented 32-bit OBJ files working with some linkers.

The assembler also is available from:

<ftp://ftp.simtel.net/pub/simtelnet/msdos/asmutl/nasm097.zip> assembler
<ftp://ftp.simtel.net/pub/simtelnet/msdos/asmutl/nasm097d.zip> docs
<ftp://ftp.simtel.net/pub/simtelnet/msdos/asmutl/nasm097s.zip> source

24.9 GAS, GNU Assembler v2.8?

Highest Processor: unknown
Limitations: unknown
Supported: <http://nasm.dhs.org/> [broken]
Platform: AIX 386BSD, NetBSD, BSDI/386, Linux, SCO, Unixware,
DOS/DJGPP
Output: unknown
Type: unknown
Comments: Little if no error control

This assembler with many object-file utilities will run on 386 systems running the following operating systems: AIX 386BSD, NetBSD, BSDI/386,

Linux, SCO, Unixware, DOS/DJGPP. The below file is a gzipped tar file. You will need gzip and tar programs to uncompress and extract the files. The assembler and utilities are part of the GNU binutils file.

<ftp://prep.ai.mit.edu/pub/gnu/binutils/binutils-2.8.tar.gz> 5018

Kb

<ftp://prep.ai.mit.edu/pub/gnu/binutils/binutils-2.8-2.8.1-patch.gz> 36

Kb

24.10 REAL TOOLS 1.0 (BETA), RTOOLS.ZIP

Highest Processor: 80486

Limitations: beta

Supported: unknown

Platform: unknown

Output: unknown

Type: unknown

Comments:

This assembler is dated in Dec 93 and is a beta test. The nice thing about this assembler is that it comes with its own DOS-windowing IDE. This assembler was written by International Systems development. The instruction set supported is 486 including protected mode instructions, but some holes do exist. This assembler has a unique way of supporting macros. 32-bit supported. On line help and debugger are available with registered product.

<ftp://ftp.simtel.net/pub/simtelnet/msdos/asmutl/rtools.zip>

24.11 GENERAL ASSEMBLER, GASM01G.ZIP v1.0f?

Highest Processor: unknown

Limitations: unknown

Supported: unknown

Platform: unknown

Output: DOS (.COM)

Type: unknown

Comments:

This is a new assembler written by Jim Gage. This version outputs .COM files and can be used to write device drivers. Another version supporting up to the 486 instruction set and .obj output is in the works. This assembler is available:

<http://www.engr.uark.edu/~jrg/gasm/gasm01f.zip> (broken)

24.12 CROSS FIRE ASSEMBLER v5.10?

Highest Processor: Pentium Family

Limitations: unknown

Supported: unknown

Platform: unknown
Output: DOS (.COM/.EXE/.SYS)
Type: unknown
Comments:

This assembler is an 80x86 assembler that uses 680x0 syntax. If you are coming from the 680x0 environment, you may want to try this as your first assembler. This assembler supports up to the pentium instruction set, 16 and 32 bit segments, supports direct generation of .com, .exe, ..sys, and more file formats, and supports pmode programming. This package comes with its own pmode DOS extender by TRAN. Currently, the math coprocessor, MMX instructions and .obj output is not supported.

You can get this assembler:

<ftp://www.simtel.net/pub/simtelnet/msdos/asmutl/xfire510.zip> (unknown)

24.13 JAS Assembler (DJGPP ASM) v1.3?

Highest Processor: unknown
Limitations: unknown
Supported: unknown
Platform: unknown
Output: unknown
Type: unknown
Comments:

Nicola Gaggi has written an assembler for DJGPP that is based upon NASM. Jas has a syntax much like TASM and is faster because it is a one pass assembler.

Download it from:

<ftp://teeri.oulu.fi/pub/msdos/programming/djgpp2/jas13.zip> (unknown)

24.14 Rodrigo Augusto's IASM v1.0

Highest Processor: Pentium Family
Limitations: unknown
Supported: unknown
Platform: independant?
Output: (.COM)
Type: unknown
Comments:

The Intel Architecture Assembler v1.0 is a platform independent assembler developed for the Intel 80x86 family of microprocessors. It has a simple syntax. The assembler was developed to get an easy to use flat memory assembler. A linker is not necessary as the assembler outputs a .COM file, but this can be changed. IASM supports instructions from all the Intel family, from the 8086/8088 to the Pentium II; MMX and floating point instructions also are supported.

IASM can generate both 16 and 32–bits code.

The assembler is available from Rodrigo Augusto's home page:

<http://www.dcc.ufmg.br/~augusto/project/> (broken)

24.15 The Visual Assembler

Highest Processor: unknown

Limitations: unknown

Supported: unknown

Platform: Win32

Output: unknown

Type: unknown

Comments: looks like it might have potential

This assembler currently is under development, but it should be worth watching. It is an attempt to apply Rapid Application Development techniques to assembly language programming. The Visual Assembler is being developed based that assembly language can be used quickly and easily to program Win32 applications though the careful implementation and use of reusable class modules rather than classes.

The Visual Assembler is being build around an IDE that will make extensive use wizard modules that will guide the user through creating Win32 applications, libraries, drivers and VxDs. The IDE will have integrated tools including a debugger, calculator, binary editor, and disassembler. The IDE will support assembling through linking to the final program.

The home page of this effort is:

<http://www.fortunecity.com/skyscraper/lycos/403/>

24.16 Gareth Owen's GASM

Highest Processor: unknown

Limitations: unknown

Supported: unknown

Platform: Win32

Output: unknown

Type: freeware

Comments:

<http://www.gaztek.org/gasm/index.html>

Use syntax similar to NASM

24.17 NBASM: The Newbasic Assembler v00.25.75

Highest Processor: Pentium Family

Limitations: beta

Supported: <http://www.cybertrails.com/~fys/newbasic.htm>

Platform: DOS

Output: DOS (.COM\,BIN)

Type: freeware

Comments: fully supported and is a great learning tool

NBASM was designed to be easy to use and doesn't need a lot of command line arguments. NBASM outputs DOS 16-bit and 32-bit code for .COM and .SYS file formats as well as the 16-bit .OBJ format, including a linker (NBL) ready for these .OBJ files.

The advantage of NBASM, it is always being updated and worked on by requests from its users, including its main user, me :).

The disadvantage, it is always being updated and worked on :) It doesn't support all of the most recent instructions, though this version now contains a lot more of them as well as some that I missed before. However, NBASM is very easy to learn and is fully supported by the author.

You can get the latest version at:

<http://www.cybertrails.com/~fys/newbasic.htm>

or directly at:

<http://www.cybertrails.com/~fys/zips/nbasm.zip>

Join the mailling/announcement list:

<mailto:nbasm-subscribe@yahoogroups.com>

24.18 FASM: The Flat Assembler v1.51

Highest Processor: Pentium Family

Limitations: ?

Supported: <http://flatassembler.net/>

Platform: DOS, Linux, Win32

Output: .BIN, MZ, PE, and COFF

Type: freeware

Comments: A very fast assembler

The flat assembler is a fast and efficient self-assembling 80x86 assembler for DOS, Windows and Linux operating systems. Currently it supports all 8086–80486/Pentium instructions with MMX, SSE, SSE2, SSE3 and 3DNow! extensions, can produce output in binary, MZ, PE or COFF format. It includes the powerful but easy to use macroinstruction support and does multiple passes to optimize the instruction codes for size. The flat assembler is self-compilable and the full source code is included.

The only difference between the various flat assembler packages is the operating system on which they can be executed. From given source each version will generate exactly the same output file, so with each of the following releases you can compile programs for any operating system.

You can get the latest version at:

<http://flatassembler.net/>

24.19 GOASM: v0.48

Highest Processor: Pentium Family

Limitations: ?

Supported: <http://www.godevtool.com/>

Platform: Win32

Output: Win32

Type: freeware

C