COMMODORE 128 PROGRAMMING SECRETS pdf

1: Retro Roms: Reading Room: Commodore Books

EMBED (for www.enganchecubano.com hosted blogs and www.enganchecubano.com item tags).

Machine Language for the Commodore 64 and Date: It taught me machine language. But it was actually Jim Butterfield. I was inspired to make a few notes about this, and how I spiraled down the rabbit hole of software development There were some cartridges available, and I had about 4 of them. I spent many an hour playing "Buck Rogers: Despite not having a storage medium, one could get programs by typing them in from a magazine. So I would do that, and it was painstaking. A game written for the Vic 20 and Commodore 64 called React. You ran around the screen picking up random objects, leaving a trail behind. I think I bought a disk drive with the proceeds. Don Whitaker The typing was laborious. And I had to leave the computer on to keep it from forgetting the program. But it was a chance to see how programs were written. SYS When I later got a disk drive, some of the games seemed to be incredibly complicated by comparison. I wanted to "list" them to see how they worked, but nothing would show. There would be some line like SYS that was all. But the files were clearly much bigger, and took a long time to load. Where was the code? It taught Machine Language at the lowest level At first I thought this was a breakthrough. Now I understood it all: I could load the programs up and see what they were doing. When I wrote my first program that actually did something it was a screen editor that helped you make stuff that looked like ANSI art I was very excited. But I quickly hit a plateau of what I could accomplish, and the code for the commercial programs I looked at was nearly impossible to read. Programming in Machine Language had simply too much bookkeeping that needed to be done There were problems similar to dealing with line numbering, where you would sometimes run out of space But this was quite a while before the Internet or StackOverflow Imagine how mad I was--years later--when I learned about assemblers, compilers, and cross-compilers! Funny Interview Story I had a job interview where we had been talking for a while. Do you like to program? He followed along with it, and we reached a point something like this: Yes--I see then the value would be in the register--that would fix it. But, what is your point? Jim Butterfield The man whose book taught me machine language apparently died in A personal webpage still seems to be up for the moment, still framed in the first person, at http: It outlines a talk he gave at York University in , two years before his death, and has this photo: Regarding potential copyright claims from his ghost, see his disappointed remark in the talk below: At that time, Dr. Dobbs was conceived as a public domain vehicle; all material in it was free from copyright. The publication still exists, but the whimsical name has been truncated to "Dr. Telling that I would have learned machine language from a good and honest communicator! Hopefully he is sipping tropical drinks with 0s and 1s on a beach in Digital Heaven somewhere. Yet puzzlingly, I am the top Google rank hit for that strangely rare text at time of writing. Perhaps I should jump in and preserve more when I see a chance to. The following material is speaking notes for a presentation I made at York University on Friday March 18th, Since they are speaking notes, they are not exactly in publication form; and, at the presentation, I might well have omitted some material and added other stuff especially in response to questions. Information on the York University Museum for early microcomputers can be obtained from: My first encounters with computers took place back in My background was math and electronics, working mostly in the field of telecommunications. My first machine was a computer nobody had heard of: So when microcomputers started to arrive in , I had an advantage over most hobbyists: I was in a good position to write about these devices, which were unfamiliar to most readers. And I did so. Most of my experience was with Commodore computers. It was a fun time. I should mention that I have never been an employee of Commodore, or owned stock in the company. My viewpoint is that of an outsider, although Commodore personnel have always been open and frank in responding to the many questions I have asked over the years. Industry prologue By , transistors had replaced vacuum tubes, and the industry had settled into a form of stability. Although there were a number of computer manufacturers, IBM enjoyed over two-thirds of the market. In part, it was because they had been in the data processing business long before the arrival of computers. Using Hollerith punched cards, IBM had an array of "unit record" equipment such as keypunch devices, tabulators, and sorters; they had decades of experience in handling data.

And with the lease came support, with system engineers on site or on call. Industry people talked about being sheltered by the "IBM umbrella. Assemblers, compilers, report generators IBM would do it free. Or, at least, at no extra charge. Underground fun Even in those days, computer programmers and operators would have secret amusements. Pictures were drawn on the line printer, patterns generated on punched card or paper tape, games were created and played, jokes were being played on co-workers, and music was being played on these computers. The seemingly impossible job of playing music on computers that had no speakers was accomplished in several ways. The hammers of line printers could be carefully timed to produce sounds of a selected pitch; the paper-advance chain could be declutched and made to furnish drum rhythms. And a transistor radio placed adjacent to the CPU, where it would pick up the electromagnetic emissions, would play a selection of popular numbers. Transitions In the time frame between and , there were a number of changes that helped shape the nature of microcomputers-to-come. In the early days, magnetic core memory was a major cost impediment, and attempts were made to circumvent it with other resources. I recall that the PDP-8S used a serial memory a mercury column delay line. It was said that if you stamped on the floor, you could change its memory contents. In , General Electric introduced "Time-Sharing" service, where users could concurrently make use of a central computer. The major impact of time-sharing on the future microcomputers was its choice of language: Basic, both adored and vilified. By , the expensive and labour-intensive magnetic core memory that had been the heart of computers started to be replaced by semiconductor memory. This set the stage for ongoing price reductions, which we still see today. This, in turn, spawned another product that was to become important in the future microcomputer world: Magnetic core memory had been non-volatile: Semiconductor memory needed to be reloaded, and the 8-inch floppy disk was created by IBM for this purpose. Initially, it was a read-only device, whose contents would be created at the IBM production facility. Viatron , then Intel As fabrication techniques were advanced, more and more elements could be packed onto an integrated circuit chip. The first chips were flip-flops and gates. The first microcomputer that I know of was made in by an almost-forgotten company called Viatron. But their fabrication plant had poor chip yields, and eventually they disappeared from the scene. The Viatron era called for some technical innovation. CRT display devices were rare and generally costly; modestly priced printers were virtually unknown. Intel entered the microcomputer field in with the But once the had been devised, the electronics industry accepted it quickly as a general-purpose component which replaced wiring with code. Motorola announced their chip very quickly. The emergence of "hobby" micros; early user groups There was a rush to build. Even before computers were offered in kit form, hobbyists were salvaging parts and building logic devices. Kits came on the market from various small entrepreneurs. Electronic houses produced their own versions, which consisted of a circuit board and a bunch of chips loose in a plastic bag. Sometimes the supplied circuit boards had printed circuit connections; other times you were expected to make the connections yourself using wire wrap techniques rarely soldering.

2: Downloads - Everything Commodore

Story time just got better with Prime Book Box, a subscription that delivers hand-picked children's books every 1, 2, or 3 months â€" at 40% off List Price.

Programs I have implemented for the Commodore Power Assembler version 8. If you know why, please tell me! Bug in Power Assembler version 8. When I assembled using the. BAS pseudo-op, I wanted some data defined using. It should be possible to do this using. Double-Ass another assembler for the C including brief documentation, this one I have not tried myself but it seems less powerful than Power Assembler Memory management Both the Commodore 64 and the Commodore have a bit address bus and can therefore address 64 kB of memory at the same time. The Commodore has more sophisticated memory management than the Commodore Using the MMU, it is possible to set four different pre-set memory configurations. If you try changing one of these three pre-set memory configurations, BASIC may no longer work correctly you may end up in the machine code monitor. The Commodore has two RAM banks bank 0 and bank 1 , each consisting of 64 kB RAM, and the memory configuration also defines which of the two RAM banks that should be switched in the other one is switched out. If memory is shared, it can be defined how much memory that is shared and if memory should be shared at the bottom or at the top of the banks or both. Sharing memory means that if bank 1 is switched in and a memory location in a shared area is accessed, it is the memory location in bank 0 that is accessed instead of in bank 1. This means that if the software tries to access a memory location in these two pages, the access is redirected to a memory location where the lower 8 address bits are the same but the higher 8 address bits are set to another value that has been defined by the user in a register in the MMU. Page 0 contains a lot of variables used by the operating system and page 1 contains the processor stack so the possibility to redirect these pages makes it possible to quickly switch between different sets of variable values and processor stacks. Of course, the memory map for the Commodore is different from that for the Commodore On the C, there are some areas that are particularly suitable for machine code programs. These are some of the major differences in the memory map. These two extra registers are used for new keys on the keyboard more columns in the keyboard matrix and 2 MHz mode. These features are described at other places on this page. It might seem a bit strange that the 2 MHz register is located in the VIC chip considering that the VIC chip does not support 2 MHz mode it results in garbage on the screen but this is the way it is. The two versions of the VIC chip are not interchangeable. With the VIC chip, you can see vertical lines going through the middle of each of the columns of the screen. When I first used a C, I thought that there was something wrong with it but I then found out that this bug exists on all Cs. However, it is more visible on some Cs than on others. This depends on the revision of the VIC chip. The bug also to a certain degree exists with the VIC chip in the C64 but there it is much less visible. Another bug in the VIC chip is that when using raster interrupts, there can sometimes be lots of white dots flashing around. This is visible in some games. The just mentioned bugs in the VIC chip can be pretty annoying and therefore if you want to play C64 games a lot, I would recommend using a real Commodore 64 rather than a Commodore in Commodore 64 mode. So, what about graphics features? The answer to this question was long believed to be no. It was found that as long as this bit is set, one raster line is skipped per clock cycle. The test bit can also be used for more advanced techniques as described in the following sections. On PAL systems, skipping an odd number of raster lines not only increases the screen refresh frequency but also leads to new colours in addition to the 16 usual ones. This is used in the C demo "Risen from Oblivion". It is impressive to see the frogs in lots of different green colours in that demo. It is the best C demo there is and one of the very few there is as well. In , it was found that skipping raster lines makes it possible to accomplish interlace mode, which doubles the vertical resolution, although it causes some flickering. The test bit tricks all have in common that they only work with real Cs, i. The VIC part of "Risen from Oblivion" for example works with most Commodore monitors I have however seen one person reporting that it did not work with his monitor. VIC graphics and character programming One difference between the Commodore 64 and the Commodore when it comes to graphics is that the C has built-in support for split-screen mode. When split-screen mode is enabled, it means that there is a vertical division of the screen into two different areas. One of the areas uses bit-map mode or multi-colour bit-map mode while the other area uses standard character mode. This is possible to do also on the C64 but there is no built-in support for it so you need to write your own raster-interrupt code to do it there. The screen editor which is a part of the Kernal in the Commodore is raster interrupt-driven in order to make split-screen mode possible. This is a difference compared to the Commodore 64 where the Kernal uses timer interrupts instead of raster interrupts. The C screen editor reads the value of so called shadow registers to update actual registers when raster interrupts occur. If they are written to directly the written value will just be overwritten by the screen editor when the next raster interrupt occurs. Instead, shadow registers should be written to. Let us for example assume that we have a split-screen with standard or multi-colour bit-map mode at the top of the screen and standard character mode at the bottom of the screen. Then, we will get two raster interrupts per screen update. We will get the "normal" interrupt at the top of the screen that we always get but since we are in split-screen mode, the raster compare register is reprogrammed when the interrupt occurs so that we will also get an interrupt further down on the screen where we want the split to occur. As mentioned in the previous paragraph, there is one "normal" interrupt occuring once per screen update and if split-screen mode is enabled, a second interrupt occurs as well further down on the screen during each screen update. However, it is only during the "normal" interrupt that the Kernal operations occuring at an IRQ are done e. If that bit is equal to 1, it is a "normal" interrupt, otherwise not. Then, you can program graphics in exactly the same way as on the Commodore 64, i. The registers have the same addresses as on the Commodore This makes it possible to quickly change the colours of a whole screen. I have come up with this idea for improvement myself. Note that this feature is only possible in C mode. The Commodore does not have this limitation. The C64 does not have pre-defined values for the sprite pointers. The sprite pointers are in the C64 and the C always located as the last 8 bytes of the 1 kB chunk of screen memory. In contrast to what many people believe, most Commodore s contain the older version of the SID chip called My two Cs both contain that version and also other people I have been in contact with who have checked the SID chip version of their Cs have that version. The only C model I know of that contains the newer version called is the Commodore DCR the metal case model that was mainly sold in North America. However, it is possible that late manufactured versions of other C models might also contain the version although I have not encountered any so far. The two versions of the SID chip are not interchangeable without adding or removing other electrical components due to different voltages. Kernal The same 39 Kernal calls as on the Commodore 64 are also available on the Commodore A few of these calls differ from the calls made on the Commodore According to "Das C Buch", it was necessary to make these changes considering some special features of the C, e. In addition to the 39 Kernal calls that also exist on the Commodore 64, there are 19 new Kernal calls that are Commodore specific. There are for example calls for accessing or jumping to memory locations in a selectable RAM bank the C has two RAM banks each consisting of 64 kB as mentioned before , for going to C64 mode, for booting from an autostart floppy, for switching between column and column modes, for programming function keys and for outputting a string of data. Some of the screen editor routines are called when the Esc key followed by another key is pressed but the routines can also be called directly from an assembly program. The routines are not described in "Das C Buch" but there are various other books that describe at least some of the routines. RS The main difference between programming the User Port RS interface device number 2 on the C compared to the C64 is that on the C a fixed range of memory addresses is always reserved for the byte output and input buffers. On the C64, on the other hand, the buffers are allocated in the end of BASIC text memory when an RS channel is opened and de-allocated when it is closed. This is not the case on the C The memory addresses used for RS system variables in zero-page are the same on the C as on the C However, the RS system variables that are not placed in zero-page e. However, when you use more than one raster interrupt per screen update you probably only want to jump to the Kernal interrupt routine once per screen update. Otherwise, the parts of the Kernal that are handled at interrupt level will be handled too often resulting in strange effects such as the cursor blinking faster than usual. Therefore, there is sometimes a need to be able to return from an interrupt without jumping to the Kernal interrupt routine. The following code should be used for this: For the Commodore 64, these lines should not be present. What the two red lines do is to load a byte from the stack and then store it in the MMU configuration register. This byte was read from the MMU configuration register and stored on the stack by the Kernal when the interrupt occurred. I have found that returning from an interrupt like explained above does not work together with BASIC you end up in the machine code monitor or the computer freezes so do this only when you are programming in assembly! However, as written above, this leads to strange effects if you have more than one interrupt per screen update sprites will move too quickly, music will play too quickly, the cursor will blink too quickly etc. If you know how to solve this, please tell me! The coding of which key that has been pressed is not the same for the Commodore as for the Commodore The value for "no key pressed" also differs. Furthermore, the Commodore has more keys than the Commodore The extra keys on the Commodore keyboard can be used also in Commodore 64 mode as is shown in this assembly code example. On the Commodore , it is possible to change the key definitions, i. Then, modify entries in the table s and change the vector s to point to the table s in RAM. Otherwise, the Kernal will just overwrite the values you have written to the vectors at the next interrupt. Since both of these books are a bit erroneous and contradictory to each other regarding this functionality, I have also had to experiment myself to see how it really works.

3: Commodore - Wikipedia

Auto Suggestions are available once you type at least 3 letters. Use up arrow (for mozilla firefox browser alt+up arrow) and down arrow (for mozilla firefox browser alt+down arrow) to review and enter to select.

None of these were present on the C64 which had only two cursor keys, requiring the use of the Shift key to move the cursor up or left. This alternate arrangement was retained on the , for use under C64 mode. A keypad was requested by many C64 owners who spent long hours entering machine language type-in programs. The two processors cannot run concurrently, thus the C is not a multiprocessing system. The shield was equipped with fingers that contacted the tops of the major chips, ostensibly causing the shield to act as a large heat sink. A combination of poor contact between the shield and the chips, the inherently limited heat conductivity of plastic chip packages, as well as the relatively poor thermal conductivity of the shield itself, resulted in overheating and failure in some cases. The SID sound chip is particularly vulnerable in this respect. The most common remedy is to remove the shield, which Commodore had added late in development in order to comply with FCC radio-frequency regulations. The C has three operating modes. C64 Mode is nearly percent compatible with the earlier computer. Selection of these modes is implemented via the Z80 chip. Based on these conditions, it will switch to the appropriate mode of operation. A sprite editor and machine language monitor were added. The screen-editor part of the Kernal was further improved to support an insert mode and other features accessed through ESC-key combinations, as well as a rudimentary windowing feature, and was relocated to a separate ROM. In column mode the editor takes advantage of VDC features to provide blinking and underlined text, activated through escape codes , in addition to the standard Commodore reverse text. A programmer with both a composite and RGB display can use one of the screens as a "scratchpad" or for rudimentary multiple buffer support. The active display can be switched with ESC-X. The back of the Commodore The VDC chip is largely useless for gaming since it has no sprites or raster interrupts. Two new disk drives were introduced in conjunction with the C A dual-disk model was announced but never produced. Later on, the 3. All of these drives are more reliable than the and promise much better performance via a new "burst mode" feature. The drive also has more on-board RAM than its predecessors, making it possible to open a larger number of files at one time. In addition, the C introduces auto-booting of disk software, a feature standard on most personal computers, but absent from Commodore machines up to that point. If the user switches to C64 mode by typing "GO 64", the drive remains in native mode, but if C64 mode is activated by holding the Commodore key down on power-up, the goes into mode which is necessary for software that performs low-level drive access. These commands are holdovers from the BASIC interpreter intended for a planned but never-produced LCD portable computer and had been intended to exit from the BASIC interpreter and to ignore keyboard input during sensitive program execution, respectively. The diskette was included with the computer, which did not include a disk drive. Software had to be made available on Commodore-specific disks formatted using the GCR encoding scheme. In addition, the cartridges only work on early model C64s from and are incompatible with later units. If one is not detected, control is passed to the and C native mode is started. If this happens, it will default to a gray background with brown text. After the kernal routine is finished executing, control is passed back to the Z C64 mode[edit] Photo from the s showing a C set-up with two disk drives and two monitors displaying the independent and column screens. Many users continued to use the inherited from their C64 system as a second drive. The C64 mode can be accessed in one of three ways: C native-mode cartridges are recognized and started by the kernal polling defined locations in the memory map. C64 mode almost exactly duplicates the features of a hardware C The extended keys of the C keyboard may be read from machine language, although the kernal routines only recognize the keys that exist on the C International models of the C use the unmodified C64 font in both modes, since the second half of the character ROM is instead dedicated to the international font containing such things as accented characters or German umlauts. This memory-mapped register, unused in the C64, determines the system clock rate. Since this register is fully functional in C64 mode, an inadvertent write can scramble the column display by switching the CPU over to 2â€”MHz, at which clock rate the VIC-II video processor cannot produce a coherent display. Fortunately, few programs suffer from this flaw. By using the higher clock rate during the vertical blank period, standard video display is maintained while increasing overall execution speed by about 20 percent. Another feature of the memory management unit is to allow relocation of zero page and the stack. Therefore, if the MMU is programmed to access blocks 2 or 3, all that results is a mirror of the RAM in blocks 0 and 1. Called the Commodore D, this new European model features a plastic chassis with a carrying handle on the side, incorporates a disk drive into the main chassis, replaces the built-in keyboard with a detachable one, and adds a cooling fan. The keyboard features two folding legs for changing the typing angle. According to Bil Herd , head of the Hardware Team a. Working to release two models at the same time had increased the risk for on-time delivery and was apparent in that the main PCB has large holes in critical sections to support the CD case and the normal case concurrently. The CDCR mounting provision is for a 60mm fan. A significant improvement introduced with the DCR model was the replacement of the video display controller VDC with the more technically advanced VDC and equipping it with 64 kilobytes of video RAMâ€”the maximum amount addressable by the device. The four-fold increase in video RAM over that installed in the "flat" C made it possible, among other things, to maintain multiple text screens in support of a true windowing system, or generate higher-resolution graphics with a more flexible color palette. Little commercial software took advantage of these possibilities. Market performance[edit] By January Info reported that "All of those rumors about the imminent death of the C may have some basis in fact". Stating that Commodore wanted to divert resources to increasing 64C production and its PC clones, the magazine stated that "The latest word online is that the last C will roll off the lines in December of ". The War Begins from Interstel had separate versions, and took advantage of column display on the C The vast majority of games simply ran in C64 mode. By contrast, many C64 productivity software titles were ported to the C, including the popular PaperClip and Paperback Writer series. The C was certainly a better business machine than the C64, but not really a better gaming machine, and people who wanted business machines bought IBM PC clones almost exclusively by the time the C was released. The main reason that the C still sold fairly well was probably that it was a much better machine for hobbyist programming than the C64, as well as being a natural follow-on model to owners with significant investments in C64 peripherals.

4: C Programming / Downloads - Everything Commodore

Search the history of over billion web pages on the Internet.

5: Commodore assembly programming - commodorese - Datorernas folkvagn

Note: Citations are based on reference standards. However, formatting rules can vary widely between applications and fields of interest or study. The specific requirements or preferences of your reviewing publisher, classroom teacher, institution or organization should be applied.

6: SID Player - H2Obsession

BASIC PROGRAMMING BOOKS AND ARTICLES Here you will find known sources for programming your Commodore computer. Commodore Programming Secrets.

7: COMMODORE BASIC BOOKS AND ARTICLES

Online shopping for Books from a great selection of Software Design, Testing & Engineering, Introductory & Beginning, Graphics & Multimedia, Microsoft Programming & more at everyday low prices.

8: Commodore Programmer's Reference Guide (PDF) | www.enganchecubano.com

COMMODORE 128 PROGRAMMING SECRETS pdf

C Programming. View Commodore Book 3 - Tips and Tricks: August mhoney MB Not rated Commodore Book 4 - Internals: August

9: history : Machine Language for the Commodore 64 and

Lou Sander's Tips and Tricks for Commodore Computers (Windcrest) Over hints and bits of advice from Louis F. Sander's famous Commodore Magazine column "Tips & Tricks". From simple hardware and programming tips to machine language programming to designing your own computer room!

COMMODORE 128 PROGRAMMING SECRETS pdf

The Fuzzy What-Was-He Ordinary differential equations with numerical techniques Bastien Aubrey Dimitri Broquard Scenes from the sex war Final fantasy x 2 piggyback guide Pain Relief for Life Rs aggarwal class 7 maths book After the fact volume 2 6th edition The magnificent Burgon, Doughty champion and defender of the Byzantine text Edward F. Hills A periodical menance to equitable principles. Forensic science advanced investigations teachers edition Trade Remedies for Global Companies Selective school practice test The big book of health and fitness Money changes everything how finance made civilization possible The judo handbook The teacher as group leader Business analysis body of knowledge Call to greatness. Minoan architectural design English vocabulary list meaning History of the Huguenot emigration to America 7. The Meo Poeple, Syed Jamal Jaafar, 61 At the dawn of glasnost Books nook World of George Orwell Tempestuous Sands Creation of form Lucy Hogan Design of shallow foundations french English architecture through the ages What Color is Your Parachute? 1994 Masters choice, volume II Philadelphia and Reading Railroad. Alkoxo and Aryloxo Derivatives of Metals Training and promotion of virtual reference services. The typicality of Oliver Wendell Holmes, by E. Carter. Learning japanese kanji practice book The ghosts of the Melting Pot Restaurant Nellies Promise (American Girls Collection) The Oxford Companion to the Photograph (Oxford Companion To.)