THE SNUGLETTER

FROM THE SOUTHERN NEVADA USERS' GROUP

Vol. 5 - No. 5 May, 1987

NEXT MEETING

MONDAY, MAY 11, 1987 - 6:30 PM

CHARLESTON PLAZA LIBRARY MEETING ROOM

PRESIDENT'S MESSAGE

By the time you read this, the TI FEST-WEST will be less than one week away! This is the closest TI show to Las Vegas. I have been to both of the previous "FESTS" and have enjoyed them both very much. As most of you are aware, Gordon Leonard has once again volunteered to drive some of us to L.A. for the show. There are still seats available in his van if anyone else is interested in going. Please contact either Gordon or myself as soon as possible if you would like to go with us. This is your chance to see all the newest hardware and software for our little orphan computer. Don't miss it!

At this month's meeting, we will be having a demonstration of one of the first drawing utilities available for the '4A. It is called Super Sketch. This drawing tablet does some remarkable things considering that it is designed to work on a bare console. It has many of the features of the more sophisticated drawing programs that came along later with the advantage that you don't even need a disk drive or memory expansion.

We will also be having a discussion and a vote on a proposition made to us recently by another user's group. This proposition could have some long-term beneficial results for our group if we accept it. It will mean that we will all have to pitch in a little, but the good it does for us will be worth it. I don't want to make the proposal public until the SNUG membership has had a chance to evaluate and vote on the proposal. Please plan on attending as the decision we make at this meeting will most likely affect our little group for some time to come.

There will also be another raffle at this meeting. Bob Bieber has a list of the prizes available. He also has a questionaire for us to fill out. This is part of a national survey being done to determine just where the average TI user is today. We would appreciate your filling out the survey for us at the meeting.

A final note:

SNUG elections are just around the corner. If you would like to run for any of the offices, now is the time to let us know. Most of the current officers have been serving for over 2 years now. It seems that the only way to get out of office around here is to move away! We have had two escapees since the last election, and have had to get replacements from within the group. Please don't run off any more of us! Step foreward and take some of the responsibility for your group. Those of us who are now serving will not abandon you. We will always be available to help out. That's how we wound up with these jobs in the first place. We are genuinely interested in the welfare of SNUG. As in any organization, when only a few people take an active part in the operation and the rest just sit and watch, it is difficult to keep things new and fresh. I'm sure that most of you have something that you could contribute if only you were asked. Well, I'm asking. Please step foreward and volunteer to serve as an officer or committee member. Make a suggestion for a demonstration. Your help WILL be appreciated.

See ya at the meeting.

-John-

MINUTES OF THE MEETING

April 13, 1987

John Martin brought the meeting to order. The 99'er FEST-WEST was discussed at length, John again reminded the members that there was a sign-up sheet for a ride to the FEST-WEST.
There was a good turn out of 26 members and two guests. That's about 50% of the membership. The members that weren't there, again missed out on some more good demo's.
I had received a flyer from MYARC in the mail about the new MYARC GENEVE 9640 and photo-copied it for the members that had not received one. The GENEVE was also discussed along with the layaway plan that DISK ONLY SOFTWARE, P.O.BOX 4170, ROCKVILLE, MD., 20850, 1-301-369-1339, was offering to purchasers of the GENEVE. Les Merryman of L and M SYSTEMS, 2330 EAST J-8 #173, LANCASTER, CA. 93535, 1-805-948-1587, the closest distributor for MYARC, said that he would have 500 in stock, very soon. The GENEVE comes with 6 pieces of software;
1) A Cartridge Saver for putting cartridge software to disk.
2) Advanced Basic- compatible with TI Basic and Extended Basic.
3) The newest version of runtime PASCAL.
4) An 80 column Word Processor.
5) An 80 column Multiplan.
6) The MYARC Disk Operating System.
(Not to mention 640k RAM, 128 VDP RAM, 12mhz clock, PC style keyboard, a monitor port, an RGB monitor port, a mouse port, joystick port, and the ability to have 256 colors at 512x424 pixels on the RGB. The GENEVE is built on a card similar to the cards in the PEB box, so it installs in a second.)
All this for less than $500.

WHAT WAS THAT PHONE NUMBER AGAIN! I THINK I'VE JUST SOLD MYSELF ONE?

Now back to reality, whew! Bob Bieber demonstrated the MAX/RLE with the latest pictures that were received from ASGARD SOFTWARE. The pictures are in the SNUG library for your use. John Martin and Bob Sherburne demo'ed a couple of good Sector Editors, one is on your FUNNELWRITER disk, by the name of Disk-Patch and Miller's Graphics Advanced Diagnostics.
Well that's about all! See you at the meeting!
<< Dee Wellman >>

!!!

SNUGLETter MENTIONED IN OTHER NEWSLETTERS

The Newsletter exchange is putting SNUG on the TI map, thanks to the great programming ability of John Martin, our President. The February, 1987 SNUGLETter and John's READ AND PRINT article was mentioned in three newsletters and was Xeroxed and reprinted in two other newsletters.
SNUG exchanges newsletters with over 60 other TI user's groups across the nation and Canada. These newsletters are in fact the life blood of the TI community. They are chocked full of information about BASIC, EXTENDED BASIC, FORTH, ASSEMBLY LANGUAGE, c99 and other languages that our TI speaks.
In the short time that I have been involved with the SNUGLETter my knowledge of the TI has increased ten-fold. I try to review all of the Newsletters that are sent to SNUG and all I can say is WOW! The people who own and operate these machines are really something. This month's batch of Newsletters contains articles ranging from using your TI as a burgular alarm system (by Rick Lumsden, DATA BUS, March '87,Delaware Valley User's Group) to the 98c RLE DIGITIZER (by Ray Kazmer, SFV 99er TImes, March '87, San Fernando Valley User's Group) this article will allow you to make the most of the MAX/RLE program(98 cents for a "DIGITIZER" is a rather high price. Mr. Kazmer, the author should have gone to the generic section, I got mine for 79 cents.)
These Newsletters are available from Lance Wilson, he is the Newsletter Librarian. Sign them out by the month and get informed. There are so many items that are covered by these Newsletters that if I mentioned something about each article for just one month's exchange the SNUGLETter would be 40 pages long.
Another excellent article is in the April '87 edition of the Front Range 99er Computer CLub, Colorado Springs, Colorado. The article, TI-WRITER GRAPHICS, by Anne Dhein, is about designing and drawing graphics using TI-WRITER, the transliterate commands and the ASCII codes in the TI-WRITER manual. The article is just to long to reproduce for the SNUGLETter. This is a very well written and documented article, please read it if you like graphics intermingled in your text. I read it too late to have incorporated it into this newsletter. Maybe the next one?

CHECK OUT NEWSLETTERS THEY ARE INFORMATIVE AND THEY ARE FUN.

!!!

UNDERSTANDING ASSEMBLY -

a beginners point of view

by John Martin

For the last few months, several SNUG members have been getting together every Sunday to learn Assembly Language. This is a project that we have started at least twice before, but were never able to get going for several reasons. First, we had no one who knew anything substantial about ANY assembly language, let alone TI Assembler. Second, even though all the articles that we have read in other newsletters and in magazines said "You have to read the E/A manual, everything is in there!", nothing was said about translating what the manual said into english plain enough for any of us to understand. Third, and maybe most responsible, there always seemed to be another "easier" language that promised to be "nearly as fast as Assembly". I refer here to the release of FORTH and more recently the C language from Clint Pulley. Don't get me wrong, I have nothing against either language. In fact, I thoroughly enjoyed learning FORTH. I probably would have liked C too, except that I got distracted from it by the BBS duties and telecommunication in general (kind of fickle ain't I).

Early this year, I got a message on the BBS from a young man named Eric Olsen. He wanted to know if anyone in the area was programming in FORTH or ASSEMBLY. After a few notes back and forth on the board, we got together in person and showed each other some of our work. He offered to teach me ASSEMBLY and I hastily agreed. He also said that he wouldn't mind showing a few others in the group what he knew, and so the SNUG ASSEMBLY SIG was born.

At the first meeting, Eric got right to the heart of the language. He talked about the three "hardware" registers and the 16 "software" registers in the machine. I discovered that this little bit of knowledge coupled with an understanding of how to decode the "Syntax definition" that is printed in the E/A manual for each instruction has taken most of the mystery out of programming in ASSEMBLY for me. I'm just a beginner and I know that I still have a LOT to learn, but thanks to what Eric has shown me, I believe that I can now learn this language in much the same manner that I learned BASIC, EXTENDED BASIC, and FORTH.

For the rest of this article, I recommend that you locate your EDITOR ASSEMBLER manual and dust it off. I will be referring to (and I hope, translating) several portions of it. You'll just have to find something else to use for a doorstop or table leg leveler for a while.

Turn if you will to page 39. This is the beginning of the chapter entitled "General Programming Information". This page refers to the "Registers" that are used by the computer. THIS IS IMPORTANT INFORMATION. There are 3 hardware registers located within the TMS9900 processor chip. None of them are directly accessable (you can't just change them arbitrarily), but there are instructions that automatically take care of it for you.

The first register is the Program Counter (PC). What this register does is hold the address of (points to) the next instruction to be executed. This is normally going to be the next consecutive even address after the instruction being currently processed. This register is affected by several instructions however, so where it actually points depends on the current instruction. Some examples of instructions that change the PC would be JMP instructions and Branch instructions. The processor looks at the address in the PC and executes that instruction.

The second hardware register is called the Workspace Pointer (WP). This is the one that does all the really neat tricks for us. The WP contains the address of the first software register (R0) in the current workspace. The Workspace consists of 16 consecutive words of RAM. These 16 words can be located virtually anyplace in memory. When your program references one of the registers, it is relative to this address. For example, if you say LI R6,15 the computer checks the WP to find out where R0 is and then goes 6 words beyond that to get to register 6. It then places the value 15 into that memory location. The exciting thing about this method of addressing registers is that you can easily set up as many workspaces as you need and branch back and forth between them without losing or changing the information stored in them. This is called a "context switch" and can allow you to do some very interesting things. There are several ways to change the WP. These would include BLWP, LIMI, and XOP. Most of the time, the BLWP (Branch and Load Workspace Pointer) instruction is the one you would use because it stores the information of all three hardware registers into three of the software registers so that you can easily go back to the old workspace.

The third hardware register is no less important. It is called the Status Register. The Status Register is updated after every instruction. The status register

(CONTINUED)

is bit mapped and keeps track of the effect of the last instruction executed. Page 40 descibes what each bit indicates, and page 41 tells which bits are affected by each instuction. The Status Register is checked by all Jump instructions except JMP which is an absolute instruction. All the rest make a comparison of some sort to determine whether to jump or not. By looking at the chart, one can see that it would be inappropriate to try to use a JOP instruction based on the results of an A (Add words) instruction. One could, however follow an AB (Add Bytes) instruction with a JOP and get some kind of results because the AB instruction does affect the Odd Parity bit of the Status Register while the A instruction does not. Don't ask me why one would want to use this particular sequence of instructions, I'm just a beginner. The point is that what determines whether to jump or not is the Status Register and not all instructions affect all bits of it.

The next thing I would like to cover is decoding the Syntax definition of the instructions. For this, lets go to page 79. This is the first page of the first section of actual instructions, the Arithmetic Instructions. This page is reprinted as the first page of each succeeding section, so you won't have to remember where it is. Just look at the first page of whatever section you are looking in to find this information. Near the top of the page, you will see definitions of a number of abbreviations. The most important (at least from my limited experience) are gas, gad, wa, iop, and wad. It would be a good idea to memorize at least these 5 abbreviations. Doing so will save you hours of frustration later on. At the bottom of the screen there are a group of symbols that are used to graphically display the execution results. These results are printed for each instruction. By translating them, you can figure out what to expect from each instruction.

Also printed for each instruction is a chart that depicts which bits of the status registe, are affected by the instruction. By looking at the caret symbols under the chart, you can tell at a glance which bits are affected by each instruction.

Now let's look at how to decode all this information on some actual instructions . Turn to page 80. This is the description of the Add words instruction. Look at the Syntax description of the word. It says:

```
[<label>] b A b <gas>,<gad> b [<comment>]
```

Translated, this means that in field 1 there is an optional label. The square brackets indicate that it is optional. Next are one or more spaces (b) to separate the fields. Field 2 is is the mnemonic (instruction). In this case the instruction is A. Again we have a space or spaces (again indicated by the b). Now we get to the part that can make us or break us. This part is called the operand field. Notice the abbreviations gas and gad? That means that virtually any type of address is acceptable for both the source operand and the destination operand. There are many instructions that require specific types of operands for either the source, destination, or both. The next field is the comment field. This field is optional.

Let's look at another instruction. Please turn to page 85. This is the instruction AI or Add immediate. Notice that the first 2 fields look similar to the A instruction, but in the third field we have <wa>,<iop>. Looking back on page 79, we discover that wa is Workspace register Address and that iop is Immediate OPerand. That means that this instruction expects (demands is more like it) to have a register (R1, R2, etc) for its first operand and a real number (no addresses, registers, etc) for it's second operand. In this case, the first operand is the destination operand. This is typical of immediate instructions. Most other types of instructions require that the second operand be the destination operand. Looking at the execution results we see (wa) + iop => (wa). This translates to "add the immediate operand to the contents of the register and put the result back into the register.

By paying attention to the things I have mentioned here and having LOTS of patience, it is possible to teach yourself how to write ASSEMBLY LANGUAGE programs. Make sure that you check the syntax definition to find out what kind of operands are expected by the particular instruction that you are using. Be aware of the affect of the instruction on the Status Register. Look at the execution results diagram for each instruction. If you follow these simple rules, you will find that writing in ASSEMBLY is not quite as hard or frustrating as you thought. If you ignore these things, you will be spending an awful lot of time going back and forth between the Editor and the Assembler looking for Syntax Errors.

In a future article, if there is enough interest, I will discuss Addressing Modes. That is a topic in itself and deserves its own article. If you want to know more about them, there is a chapter called Addressing Modes that begins on page 56. I recommend reading it at least once. By creative use of addressing modes, there is really no limit to what you can do.

!!