

00000000h: 4E 45 53 1A 08 10 40 00 00 00 00 00 00 00 00 00
00000010h: FF FF 50 42 2D 30 20 3D 3D 3D 3D 3D 3D 3D 3D 3D
00000020h: 54 80 5F 86 D2 8A 5F 86 AF 80 A2 86 F4 A2 86 F4
00000030h: 24 80 4C 80 14 8D 34 8D 44 8D 54 8D 64 8D 74 8D
00000040h: 84 8D 94 8D A4 8D C4 8D D4 8D F4 8D 04 8E 04 8E
00000050h: 04 8E 24 8E 44 8E 5C 8E 74 8E 80 8E 88 8E 48 97
00000060h: 48 9C 08 9E 00 55 00 00 00 00 00 05 55 55 55 55
00000070h: 55 55 55 AF AF AF 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
00000080h: FD AA AB AB AA 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F
00000090h: 5F AA AF 05 05 00 00 00 00 00 00 00 00 00 00
000000a0h: 55 55 55 55 55 55 F5 FF AF AA 00 05 0A 0A 0A 0A
000000b0h: 0A F7 AE AE AB AA 0A 0A 0A 0A 55 55 05 AF AA 00
000000c0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03
000000d0h: 03 03 03 03 03 03 03 03 03 03 03 03 03 03 03 03
000000e0h: 7D 7E 7F 03 7B 02 7A 03 78 02 02 03 75 67 7E
000000f0h: 7C 03 03 02 79 7C 03 02 02 77 03 74 71 72 7D
00000100h: 7E 7C 03 7B 02 79 7C 78 02 02 77 75 76 71 72 7D
00000110h: 7E 7F 7E 7B 02 7A 02 78 02 02 02 75 76 73 74 7F
00000120h: 7E 7C 03 7A 02 79 7C 02 02 02 77 73 74 71 72 03
00000130h: 03 03 03 03 03 03 03 03 03 03 03 03 03 03 03 03
00000140h: 03 03 03 03 03 03 03 03 03 03 03 03 03 03 03 03
00000150h: 03 03 03 03 03 03 03 03 03 03 03 03 03 03 03 03
00000160h: 03 03 03 03 03 03 03 03 03 03 03 25 26 03 03 03
00000170h: 03 21 22 1F 24 2F 0D 1C 20 1B 1C 1D 01 19 1D 23
00000180h: 03 0F 10 0E 14 02 15 41 19 02 1A 01 1E 19 1B 11
00000190h: 0D 12 13 16 17 18 01 02 1C 01 1E 02 1D 01 01 27

The Theory of Relative Searching

August 22, 2009

Version
1.01

by Samuel S. Kempf



Einstein's Theory of Relativity is considered to be one of the utmost works of modern genius. Relative searching, while not quite as awe inspiring, seems to be almost as difficult to grasp for the fledgling ROM hacker. With this article, I intend to expound upon the theory of relative searching and explain the basis of how it works.

This document is not intended to explain the technical details of relative searching. Each hex editor (as well as the relative search utilities that preceded the feature's common implementation in editors) works differently, so you will need to refer to your application's documentation for instructions. What I will explain is the underlying technique that a relative search utility uses, that is, what the tool does, not how to make it do so.

Table Of Contents

1. [First Things First](#)
2. [Basic English](#)
3. [Turning Japanese](#)
4. [Graphically Inclined](#)
5. [Conclusion](#)
6. [Solutions to Exercises](#)
7. [Credits & Release Notes](#)
8. [About The Author](#)

First Things First

Before you can understand relative searching, you must first understand the basics of how fonts are stored in console games. In a document, such as this tutorial, there is a clear distinction between text and graphics. You can cut & paste the text into a basic text editor and modify it as you wish, but the images would have to be accessed with an image editor in order to manipulate them. With ROMs, there is no distinction between text and graphics. They are the exact same thing. This is why you can't simply open a ROM in a word processor and edit the text. The only reason we refer to the tiles that contain letters, numbers and punctuation as a font is for the sake of convenience.



Image #1

Image #1 is a typical 8x8 font as viewed in a tile editor. The red lines indicate the border of each tile. It is important that you understand this tile concept. When a game displays text on the screen, all it is doing is displaying these graphic tiles. To the console, the word MARIO is no different from a picture of Mario. One of the biggest roadblocks a novice hacker encounters when learning to relative search is that they are dealing with pictures, not words. If you can grasp this concept of tiles, it will be easier to understand the underlying theory of relative searching.

Basic English

To understand what relative searching is and how it works, we need to look at the first word of the term: relative. More importantly, we need to examine its root word: relate. What many people fail to understand is that a relative search does not care about the entities, it only cares about how they relate. It is the relationship between two or more items that a relative search program seeks out, not the items themselves. A relative search utility has no idea what the letters C, A or T mean, but it does know that (in standard alphabetical order) the value of the letter A is 2 less than the value of the letter C and that the value of the letter T is 19 more than that of A.

Chart #1:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Examine the chart above and you can verify that the relationships listed previously are, in fact, correct. 1 is certainly 2 less than 3, 20 is 19 more than 1. Don't get hung up on assigning the values to the letters, however. As I said before, what is important is not the values of the individual characters, but the relationship between them: the relative values. In this case, the relative value of the word CAT is -2,+19.

As has been stressed repeatedly, what is important is not the individual values of the characters, but the relationship between those values. Consider the (somewhat imaginary) word GEX. The letter E is 2 spaces to the left of the letter G and the letter X is 19 spaces to the right of E. This makes the relative value of the word GEX -2,+19, identical to the relative value of CAT.

Exercise #1: Find the relative value of the word PIE using Chart #1. ([Solution →](#))

Chart #2:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52

Let us arbitrarily assign A the value of 27. With these values, A(27) is still 2 less than C(29) and T(46) is still 19 more than A. The relative value of the word CAT (-2, +19) remains unchanged.

Exercise #2: Find the relative value of the word ROCKET using Chart #2. ([→](#))

Now, for the sake of example, let's try mixing things up a bit.

Chart #3:

C	A	G	K	E	I	O	M	S	Q	W	U	B	Y	F	D	J	H	N	L	R	P	V	T	Z	X
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89

It is unlikely that you will ever encounter a font scrambled quite like this, but you will often encounter fonts that do not have the characters in a standard order (for instance, ordered top to bottom instead of left to right), so this exercise will show you what to do regardless of what order the font may be in.

Sticking to our primary example, in this case A(65) is 1 more than C(64) and T(87) is 22 more than A, making the relative value of CAT +1,+22. What order the characters are in makes no difference, all a relative search cares about is the distance between the characters.

Exercise #3: Find the relative value of the word CALIFORNIA using Chart #3. ([→](#))

So now let's take a moment to discuss what a relative searcher does with these relative values, using our original example of CAT having a relative value of -2,+19.

It is actually quite simple. The relative search utility examines each byte of a ROM in turn and checks to see if the value of the following byte is 2 less than the value of the byte it is examining. If not, it moves on to the next byte. If that second value is 2 less than the first, however, it then checks to see if the third byte's value is 19 more than the second. If so, it marks that location down as a match and continues searching. If not, it simply moves on to the next

byte, continuing until it has reached the final byte of the ROM. It then outputs any matches it may have found so that you can do with them as you please. Using the -2,+19 example, any of the following 3byte values would produce a match: 2A283B, 030113, E7E5FA.

So how would you determine which of these values actually corresponds to the word CAT? You change the first value of the first match, save your changes in the hex editor, then open the ROM in an emulator and see if it changes the word CAT in the game. If not, undo your change, then go to the next match and try again. Barring compression, strange text encoding or a mistake in your calculation of the relative values, you will eventually find the location of the word. With this accomplished, you can build your table file and advance your project.

So far, we have dealt exclusively with capital letters. What happens if the font also includes lowercase letters? Absolutely nothing, you just deal with more values.

Chart #4:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52

The above chart expands Chart #1 to include lowercase letters. Using the same technique previously explained, the word CAT (uppercase) still has a relative value of -2,+19. The word cat (lowercase) also has a relative value of -2,+19. How can this be? Remember, relative values don't care *what* a character is, it only cares *where* it is in relation to the character before it. An uppercase A is two characters to the left of an uppercase C and a lowercase a is two characters to the left of a lowercase c, so the relative value of CA and ca is identical. Now when you deal with a mixed case word, you will find the difference between uppercase and lowercase letters to be quite large but the technique is otherwise unchanged. The word Cat (mixed case) has a relative value of +24,+19.

Exercise #4: Using Chart #4, find the relative values of the following words: **a.** SOUP
b. soup
c. Soup(→)

Now that you have mastered determining relative values using the charts above, let's try doing the same with an actual game font. The following picture (Image #2) is a screen shot of the font from the NES game The Jetsons - Cogswell's Caper! Why this game? Because I thought it only polite to use some mediocre game that would be unlikely to ever play a role in ROM hacking. What game it is is not important, however, as we will only be utilizing it for a series of mental exercises.

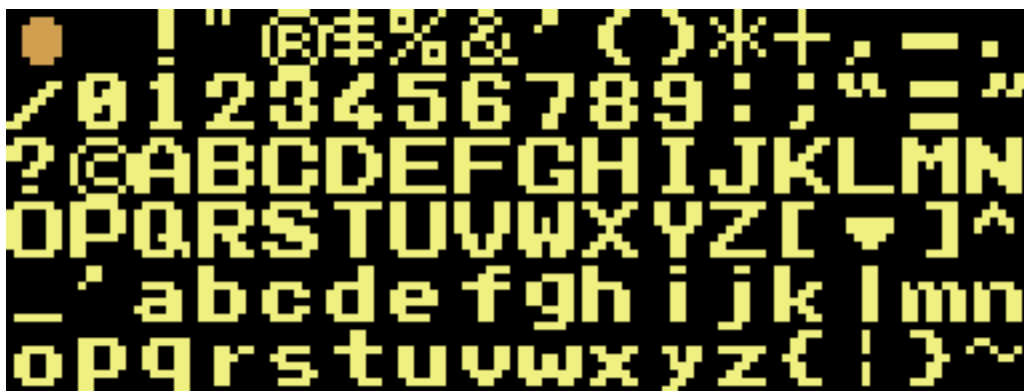


Image #2

As you can see, there are a few non-alphabetic characters between the capital Z and the lowercase a. This does not change how you determine relative values, it simply means the relative value between a capital letter and a lowercase letter will be slightly larger. In the font above, the relative value of CAT (and cat) is still -2,+19 but the relative value of the mixed case Cat is +30,+19.

Remember, *how* the characters are positioned in the graphic is unimportant, all that matters is their position relative to one another. Regardless of whether you choose to think of the ! as 1 and the capital A as 33 or ignore the preceding punctuation and designate the capital A as 1, the lowercase 'a' is still 30 characters right of the capital C and the lowercase 't' is 19 characters right of the lowercase 'a', so the relative value remains +30,+19. (**Note:** While it does not matter with which character you begin your mental numbering process, it is important that you increment your numbers from left to right and top to bottom, increasing the value by 1 each time and skipping none of the tiles between your starting and ending points. If you designate A as 1, then B is 2 and so on down to z being 58. When you reach the end of a line, continue on with the first character of the next line.)

You relative search for terms that include numbers or punctuation in the exact same manner as words. Once again, a relative search doesn't care *what* it's searching for, it only cares *where* the tiles are in relation to one another. You can even relative search for non-font graphics, which is a technique that occasionally comes in handy when hacking a title screen or when you need to re-arrange a graphic that utilizes Japanese words with repeated tiles.

Exercise #5: Using Image #2, calculate the relative value of the following words: **a.** cow
b. Taco
c. Aztec
d. InVerse
e. 12Dozen([→](#))

Most relative search utilities will allow you to use wildcards in your search. This is particularly useful when you can't make out what a particular character is, or you believe that a control code might appear within the text for which you are searching. In such a case, you would use an asterisk (*) in place of the unknown entity. For instance, say that you know the word bolögna appears somewhere in the game, yet you don't see the ö symbol anywhere in the font. You would replace the ö with an * and the relative value of the g after it would be it's difference from the l that precedes it. In other words, utilizing a wildcard will cause the relative searcher to completely ignore the value of that particular byte. The relative value of bolögna, using the font in Image #2, would be +13,-3,*,-5,+7,-13.

Exercise #6: Using Image #2, calculate the relative value of Frögger([→](#))

If you have used relative searching in the past, you may wonder why this knowledge is important. After all, most hex editors require you to input the actual word you are searching for, not it's relative value. This is all well and good when you are searching for an English word in a game that has a standard order font, but what if the font is out of order, you are translating a Japanese game or you need to rearrange a title screen? In a case like this, you won't be able to use the relative search feature that is incorporated in most hex editors. You'll need a tool that allows you to manually input the relative values yourself. As of the writing of this article, Translhexction is the only hex editor that offers this feature. Monkey Moore is an excellent standalone utility. WindHex does include a kana search feature, but this will only work with games that store their fonts in the same order as the font in WindHex.

Turning Japanese

Relative searching is extremely useful for locating Japanese text in a ROM. With over 90 characters in the kana character set, another 50+ modified characters and thousands of kanji, few games actually store their fonts in any standard order. When seeking to translate a Japanese ROM, relative searching is an absolute necessity for table building. As I have stressed repeatedly, the technique remains the same. The only difference is that it will take longer to find the characters in question if you are unfamiliar with the Japanese language.

Image #3 is a screen shot of the font from Chibi Maruko-Chan - Uki Uki Shopping, which translates (very) roughly into Little Maruko - Exciting Shopping. As far as I can tell, this is a 1-4 player board game starring Japanese school children who go on a shopping trip. I chose it because it was the first game I randomly came across that had a fairly clear font. Incidentally, the font is in a very strange order which makes it a challenge even if you are familiar with Japanese.

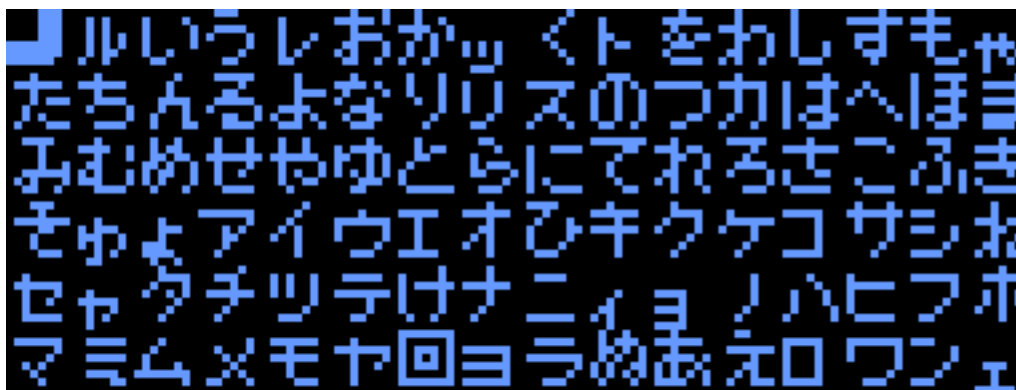


Image #3

Using the font order depicted in Image #3, the relative value of こねこ (koneko, the Japanese word for kitten) is +18,-18. (The characters used are in column 14, row 3 and column 16, row 4.)

Exercise #7: Using Image #3, find the relative value of the following words. I understand that it will be hard for those unfamiliar with Japanese to locate some of the kana, so the position of these characters will be listed after the exercise, but I suggest attempting to find them yourself first, as you won't have such a guide to help you when hacking a ROM on your own. (After awhile, however, you will become familiar with the basic kana and their pronunciation, even if you don't know what the words they form mean.) After each Japanese word is the pronunciation and the English meaning of the word.

- a. メタル (metaru / metal)
- b. かたな (katana / sword)
- c. ほんのう (hon'no / instinct)
- d. たのしむ (tanoshimu / enjoy)
- e. サマータイム (sama-taimu / daylight savings time) [Use a wildcard (*) for — as it is not in the image.]

- a. c4r6, c3r5, c2r1
- b. c7r1, c1r2, c6r2
- c. c15r2, c3r2, c10r2, c4r1
- d. c1r2, c10r2, c13r1, c2r3
- e. c14r4, c1r6, *, c3r5, c5r4, c3r6(→)

Relative searching for Japanese words may seem more difficult at first but that is simply because you are unfamiliar with the language. As you gain experience hacking Japanese ROMs, you will find it is just as easy to relative search in Japanese as it is English.

Graphically Inclined

As previously mentioned, it is also possible to locate some graphics using a relative search. The most common reason for doing so is if a game makes use of a graphic that includes text and that text repeats certain tiles rather than wasting space with multiple instances of the same graphic. This is good programming technique but increases headaches for innocent ROM hackers.

As a theoretical example, let's say you are hacking a game that contains the word こねこ (koneko) in large cartoonish letters, rather than the font used for regular text. Furthermore, rather than having こ taking up twice as much space in the ROM with two occurrences, the programmer repeated the first instance. This means that if you change the graphic, it will affect both occurrences. In order to hack the graphic to say KITTEN, you will have to determine where the graphic is called in the ROM and change the values of the second こ to unused tiles. You can't simply relative search for こねこ because, while readable text, the word does not utilize a standard font and text routines. What you must do is locate the graphic in a graphics editor such as you initially do with a font and determine the relative value of the graphic.

Let's say that each letter in the graphic is two tiles wide by two tiles tall and are separated by a space. What you will relative search for is the top line of the graphic, as there is no way of knowing what, if any, values may appear between each line. The relative value of the top line of the graphic should be +1,+1,+1,+1,-2,-2,+1. This is because the second tile of the graphic would be one space to the right of the first, followed by the aforementioned space, then the top two tiles of the second character, then jumping back two values to repeat the space, jumping back another two spaces to repeat the first tile of the first character, then also repeating the second tile of the first character. Doing a relative search for these values should allow you to locate the graphic and change the tiles of the second Σ to unused tiles containing your new graphics.

Now for a more concrete example:

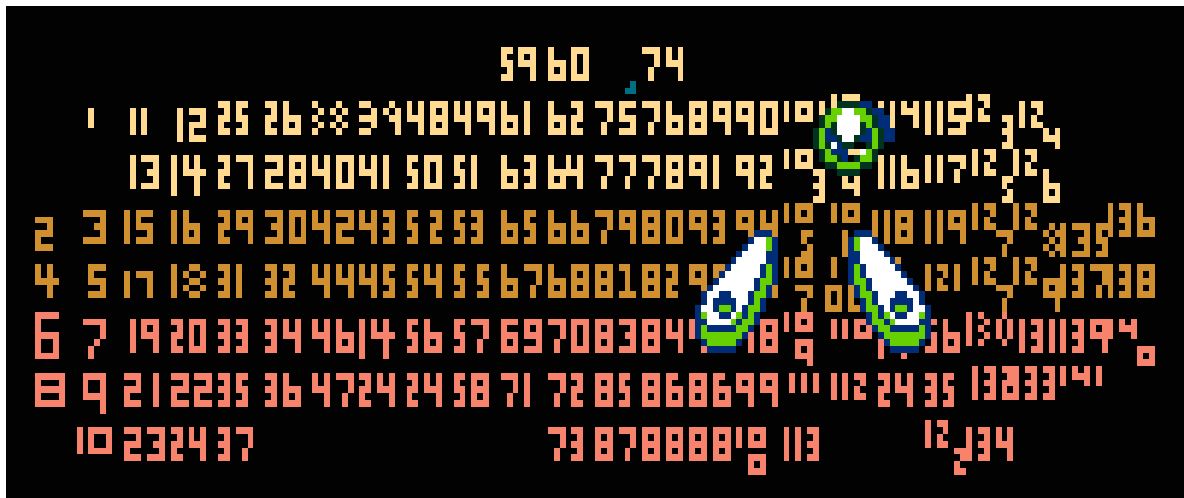


Image #4

This is a section of a hacked title screen for the game Family Pinball for the Famicom. It was hacked using the technique described in my tutorial *Title Screen Hacking Made Easy*. Take a look at the very bottom row in the middle and you'll see that the number 88 is repeated. This means that a single tile is reused. Whatever I draw in place of that tile will also be repeated, unless I locate the title screen layout in a hex editor and modify it.

Fortunately, since the numbers appear in numeric order, it is extremely easy to determine the relative value. Let's take the first five characters of the first long line (1,11,12,25,26). The relative value of this section would be +10,+1,+13,+1. Searching using this criteria produced only one result, which was the line I was looking for. I now know where the title screen layout is stored in the ROM and I can edit it as necessary.

Conclusion

Relative searching is one of the most important techniques in basic ROM hacking. It eliminates a large amount of the trial and error you would otherwise have to go through to discover the location of text in a ROM, greatly streamlining the process of creating a table file.

You should now have a strong grasp of how relative searching works. If you were having trouble successfully executing a relative search previously, you should now be able to find that elusive data. If you are already an experienced ROM hacker, hopefully you were able to glean a few new tidbits of knowledge. In either case, I hope you found this article concise and informative. If not, a full refund will be issued.

Solutions to Exercises

Exercise #1: -7,-4 ([← Return to Exercise](#))

Exercise #2: -3,-12,+8,-6,+15 ([←](#))

Exercise #3: +1,+18,-14,+9,-8,+14,-2,-13,-4 ([←](#))

Exercise #4: ([←](#))

- a. -4,+6,-5
- b. -4,+6,-5
- c. +22,+6,-5

Exercise #5: ([←](#))

- a. +12,+8
- b. +13,+2,+12
- c. +57,-6,-14,-2
- d. +37,-24,+15,+13,+1,-14
- e. +1,+18,+43,+11,-21,+9

Exercise #6: +44,*, -11,0,-2,+10 ([←](#))

Exercise #7: ([←](#))

- a. -17,-65
- b. +10,+5
- c. -12,+7,-22
- d. +9,-13,+21
- e. +19,*, -14,-14,+30

Credits & Release Notes

This text was authored by Samuel S. Kempf. The most recent version of this document, as well as other documents written by the same author, can be found at Suicidal Translations. (<http://www.suicidaltranslations.com>)

Thanks to Darth Nemesis and DarknessSavior for correcting mistakes in the Japanese language section.

Thanks to Spinner 8 for uncovering a couple of grammatical errors and one link related bug.

Thank you to BRPXQZME, Kiyoshi Aman and Black_Phantom for information about how to do hexadecimal arithmetic on non-Windows operating systems.

Special thanks to Lleu for suggestions that triggered a thought on how to better present some information contained herein as well as numerous suggestions regarding content, many of which were taken in part or in whole.

A tip of the hat to Demi, The Mad Hacker and toma for their early ROM hacking documents that contributed greatly to my initial understanding of basic ROM hacking techniques and fueled my desire to create documents of my own.

This document was created with OpenOffice.org Writer (v2.0 and 3.0) (<http://www.openoffice.org>)

All images were captured and edited with GIMP (v2.4 and 2.6) (<http://www.gimp.org>)

All hexadecimal images, both on the title screen and in the Super Secret Super Fun Time Exercise were captured from Hexecute by mdw. Hexecute isn't the best option for hex editing (I prefer WindHex) but it certainly is the prettiest of the ROM hacking specialized editors.

Music listened to during the writing of this document: *Gone Ain't Gone* by Tim Fite, *On A Clear Night* by Missy Higgins, *Urban Angel* by Natalie Walker, *The Greatest* by Cat Power and *Love in the Time of Science* by Emiliana Torrini

Image #1: Nester's Funky Bowling © 1996 Nintendo

Image #2: The Jetsons – Cogswell's Caper © 1992 Taito Corporation

Image #3: Chibi Maruko-Chan - Uki Uki Shopping © 1991 Namco Ltd.

Image #4: Family Pinball © 1989 Namco Ltd.

Images #5, #6 & #7: Super Mario Bros. © 1985 Nintendo

The Theory of Relative Searching by [Samuel S. Kempf](#) is licensed under a [Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License](#). You are free to distribute this document in any manner you choose, as long as said distribution is noncommercial in nature. Furthermore, you may add to or edit this document in any manner as long as the original author is attributed. However, the author does ask that you attempt to contact him prior to modifying this document so that any possible changes can be discussed for inclusion in future official releases from the original author. Any derivatives of this document must be released under the same Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License.

About The Author

Samuel S. Kempf (formerly known within the emulation scene as **InVerse**) has been ROM hacking for over a decade, beginning with a professional wrestling themed graphical hack of Tennis for the NES that was released in 1998. One year later, he began authoring ROM hacking tutorials, starting with the now obsolete *Constructing Tables for NES ROM Hacking*. His other ROM hacking documents include *The Definitive Guide to ROM Hacking Tables* and *The Definitive Guide to ROM Hacking for Complete Beginners*.

Under the group name of **Suicidal Translations**, Mr. Kempf has released over a dozen translation patches, for games such as Hello Kitty World, Kid Niki 3 and Labyrinth, as well as the table making utility Table Auto-Generator. He is the founder of the now defunct ROM Hacking Repository (formerly located at romhacking.com) as well as a former staff member of Zophar's Domain (www.zophar.net) and RHDN (www.romhacking.net.)

Aside from ROM hacking, his hobbies include reading (favourite authors include Jack Kerouac, Mindy L. Klasky and Malcolm Gladwell), listening to music (favourite artists include Tori Amos, Marilyn Manson and Tim Fite) and, of course, video games (of which he most favours Crystalis [NES], Mass Effect [360] and Ocarina of Time [N64])

Due to a lack of broadband availability in his rural location, Mr. Kempf rarely has a chance to play on Xbox Live, but when the opportunity to do so arises, his GamerTag is "The Dharmatist". Other contact information can be found at the [Suicidal Translations](http://www.suicidaltranslations.com) website.



Super Secret Super Fun Time Exercise

Now let's try an exercise that will truly test your understanding of relative searching. Let's do a manual relative search! You'll never need to do this when hacking a ROM unless you're trapped on a deserted island with no Internet access but somehow come up with a ROM, tile viewer and hex editor, but if you can successfully complete this exercise, you should have no problem relative searching by traditional automated means.

Here we have the font of a classic NES game:



Image #5

And here we have a cross section of this game loaded in a hex editor with no table file.

00000CD0	1630	270f	0f30	1000	3f00	200f	291a	0f0f
00000CE0	3617	0f0f	3021	0f0f	2717	0f0f	1627	180f
00000CF0	1a30	270f	1630	270f	0f36	1700	3f00	200f
00000D00	291a	090f	3c1c	0f0f	3021	1c0f	2717	1c0f
00000D10	1627	180f	1c36	170f	1630	270f	0c3c	1c00
00000D20	3f00	200f	3010	000f	3010	000f	3016	000f
00000D30	2717	000f	1627	180f	1c36	170f	1630	270f
00000D40	0030	1000	3f00	0422	3000	1000	3f00	040f
00000D50	3000	1000	3f00	0422	2716	0f00	3f14	040f
00000D60	1a30	2700	2548	101d	110a	1714	2422	181e
00000D70	2416	0a1b	1218	2b00	2548	101d	110a	1714
00000D80	2422	181e	2415	1e12	1012	2b00	25c5	160b
00000D90	1e1d	2418	1e1b	2419	1b12	170c	0e1c	1c24
00000DA0	121c	2412	1726	050f	0a17	181d	110e	1b24
00000DB0	0c0a	1c1d	150e	2b00	25a7	1322	181e	1b24
00000DC0	1a1e	0e1c	1d24	121c	2418	1f0e	1baf	0025
00000DD0	e31b	200e	2419	1b0e	1c0e	171d	2422	181e
00000DE0	240a	2417	0e20	241a	1e0e	1c1d	af00	264a
00000DF0	0d19	1e1c	1124	0b1e	1d1d	1817	240b	0026
00000E00	8811	1d18	241c	0e15	0e0c	1d24	0a24	2018
00000E10	1b15	0d00	0aa8	6885	0468	8505	c8b1	0485

Image #6

Exercise #3.14159265: Using the font in Image #5, calculate the relative value of the word PRINCESS. Now find the location of the word PRINCESS in the hex code displayed in Image #6. (Note: There will be only one result in this exercise, so when you find it, you can quit searching through the rest of the hex.)

If you are unfamiliar with how to do hexadecimal arithmetic, you are allowed to cheat and use a calculator. Windows Calculator can do this by selecting Scientific from the View menu and then selecting Hex. In MacOS, select Programmer from the View menu and then click Hex from the input mode selector. For *nix you can use SpeedCrunch for KDE (press F8 for Hex mode) or gcalctool for Gnome (Ctrl-S for scientific mode.)

This Page Intentionally Left Blank

This Page Intentionally Left Blank

Solution #3.14159265: The relative value of PRINCESS is +2,-9,+5,-11,+2,+14,+0.

The word PRINCESS begins at 0xD87, as shown in Image #8.

00000CD0	1630	270f	0f30	1000	3f00	200f	291a	0f0f	M..FF.G...WF.QFF
00000CE0	3617	0f0f	3021	0f0f	2717	0f0f	1627	180f	.NFF.XFF.NFFM.OF
00000CF0	1a30	270f	1630	270f	0f36	1700	3f00	200f	Q..FM..FF.N...WF
00000D00	291a	090f	3c1c	0f0f	3021	1c0f	2717	1c0f	.Q.F.SFF.XSF.NSF
00000D10	1627	180f	1c36	170f	1630	270f	0c3c	1c00	M.OFS.NFM..FC.S.
00000D20	3f00	200f	3010	000f	3010	000f	3016	000f	..WF.G.F.G.F.M.F
00000D30	2717	000f	1627	180f	1c36	170f	1630	270f	.N.FM.OFS.NFM..F
00000D40	0030	1000	3f00	0422	3000	1000	3f00	040f	..G....Y..G....F
00000D50	3000	1000	3f00	0422	2716	0f00	3f14	040f	..G....Y.MF..K.F
00000D60	1a30	2700	2548	101d	110a	1714	2422	181e	Q.....GTHANK YOU
00000D70	2416	0a1b	1218	2b00	2548	101d	110a	1714	MARIO!...GTHANK
00000D80	2422	181e	2415	1e12	1012	2b00	25c5	160b	YOU LUIGI!...MB
00000D90	1e1d	2418	1e1b	2419	1b12	170c	0e1c	1c24	UT OUR PRINCESS
00000DA0	121c	2412	1726	050f	0a17	181d	110e	1b24	IS IN..FANOTHER
00000DB0	0c0a	1c1d	150e	2b00	25a7	1322	181e	1b24	CASTLE!...JYOUR
00000DC0	1a1e	0e1c	1d24	121c	2418	1f0e	1baf	0025	QUEST IS OVER...
00000DD0	e31b	200e	2419	1b0e	1c0e	171d	2422	181e	.RWE PRESENT YOU
00000DE0	240a	2417	0e20	241a	1e0e	1c1d	af00	264a	A NEW QUEST....
00000DF0	0d19	1e1c	1124	0b1e	1d1d	1817	240b	0026	DPUSH BUTTON B..
00000E00	8811	1d18	241c	0e15	0e0c	1d24	0a24	2018	.HTO SELECT A WO
00000E10	1b15	0d00	0aa8	6885	0468	8505	c8b1	0485	RLD.A.....

Image #8

If you successfully completed this exercise without developing a migraine headache, you may just be destined to become the next great ROM hacker.



...the Akahana Consortium welcomes you...