Learn to program an Apple 1 – Beginner's Instructions

Follow these instructions to learn how to use an Apple 1 (Replica 1 Plus) computer. Note: These procedures can also be run using Apple 1 emulator software, such as Open Emulator.

1. Turn on the Replica 1 Plus using the switch in the upper left corner of the board. The monitor should display random characters on the screen.

2. Press the RESET button on the middle left of the Replica 1 board. You should see a flashing @ sign at the bottom of the screen.

3. Press the CLEAR button on the middle right of the Replica 1 board. The screen should clear.

4. Press the ENTER key (also known as the Return key) a few times.

5. Type the following, then press ENTER:

   300.30F

6. The computer should display 2 lines of memory. The first memory address is $0300 in hexadecimal (base 16), which corresponds to 768 in decimal (16*16*3). The second memory address is $030F. The letter F represents the number 15, so the address = 768 + 15 or 783.

7. Enter the following hex codes to store a program at address $300. If you make a mistake, press ENTER and retype the line. (Note: The underscore character "_" can be used as a backspace, but the screen will not back up and erase the previous character – it will internally delete it in memory.

   300:A9 00 AA 20 EF FF E8 8A 4C 02 03

8. The computer will respond by showing you the previous contents of address $300 before you changed it to $A9. Type the following to verify your work:

   300.30F

9. If you entered all the codes for the program correctly, try running it by typing:

   300R

10. If the program runs successfully you will see all the Ascii characters displayed in order on the monitor. The sequence will repeat until it fills the screen. Notice that there are no lower case letters. Lowercase did not become standard until the Apple //e was released in 1983. Earlier Apple][computers required special hardware for entry and display of lowercase. Now is a good time to get a photo with your first Apple 1 program running successfully!

To understand what the codes you typed mean, examine the following listing. The addresses and hex codes you typed are in the left column. The middle column contains machine language instructions that a programmer uses to tell the computer exactly what to do. The right column contains comments explaining what the instructions do.

```
                     ORG $0300   ; Enter program at address $300 (768)
300:A9 00            LDA #$00    ; Store 0 in the Accumulator register
302:AA               TAX         ; Transfer A to the X register
303:20 EF FF         JSR $FFEF   ; Call a subroutine to print the
                                 ; character in the A register.
306:E8               INX         ; Add 1 to the X register
307:8A               TXA         ; Transfer the X to the A register.
308:4C 02 03         JMP $302    ; Jump to address $302
```

Now you will learn how to write a program using Apple BASIC. The Apple 1 used a version of the BASIC language written by Steve Wozniak, which only supported Integer values – it did not have decimals. Also, the number range was limited to values between -32767 and +32767.

1. Push RESET, then CLEAR. Then press ENTER a few times.
2. Type the following command:

   E000 R
3. You should see the prompt: **>** This means that BASIC is running.
4. Press ENTER a few times.
5. Enter the following lines. Press ENTER at the end of each line. Fill in your name where it says "your name" below.

   10 X=1 : Y=1
   20 TAB(X) : PRINT "HELLO your name"
   30 X=X+Y : IF X = 10 THEN Y = -1
   40 IF X > 0 THEN 20
   50 END
6. Type: **LIST** and press ENTER to verify that you typed each line correctly.
7. Type: **RUN** and press ENTER. If the program works, you will see your name printed 19 times forming an arrow pattern on the screen. Congratulations! Take another photo. If you want to learn more, continue with the Advanced worksheet on the next page.

The Replica 1 Plus computer is sold by ReActiveMicro.com. It was designed by Vince Briel (brielcomputers.com).
*The **WozPak Special Edition** book is sold by www.CallApple.org.*
This public domain document was written by Eric Rangell (erangell@gmail.com).

Learn to program an Apple 1: Advanced Worksheet

The program that you will enter will display your name 20 times on the screen, once on each line.

As you read the instructions below, fill in the hex values for each box in this table.

```
                    ORG $0300   ; Enter program at address $300 (768)
300:                            ;
                    LDY #20     ; Number of lines to print
             OLOOP  LDX #1      ; Start with first character
             ILOOP  LDA $340,X  ; Get the ascii code for the character
                    JSR $FFEF   ; Send the character to the display
                    INX         ; Bump index to next character
                    CPX $340    ; Are at the last character?
                    BCC ILOOP   ; If X < # characters, repeat loop
                    BEQ ILOOP   ; If X = # characters, repeat loop
                    LDA #$8D    ; Load A with Carriage Return
                    JSR $FFEF   ; Send it to the display
                    DEY         ; Count down # of lines
                    BNE OLOOP   ; If not done, repeat outer loop
                    RTS         ; Return to caller of this subroutine
                    JMP $FF1F             Apple 1 Monitor.
340:05 C8 C5 CC CC CF           ; Data to print: 5 characters, HELLO
```

TODO: Add the Ascii character for the SPACE after the $CF (the letter O in HELLO), then add the Ascii characters for your name (see last page of this document), then change the value of $340 from 5 to the hex value of the total number of characters you want to print.

1. Hexadecimal numbers are base 16 - they are used for computer programming because it is very easy to convert them to binary. The hex digit $F = decimal 15 = binary 1111. The number 16 requires 2 hex digits: 1 for the number of 16's, and 0 for the remainder of 0. The decimal number 17 = hex $11.

2. Computer machine instructions are coded with hex numbers based on the instruction name and the addressing mode (which tells the computer how to find the data for the instruction). See the opcode chart (see next to last page of this document) for the hex values for each 6502 instruction.

3. Convert the decimal number 20 to hex. Look up the opcode for LDY for the Immediate addressing mode. Enter the opcode in the first box and the hex value for 20 in the 2nd box. The LDY instruction LOADS the Y register of the 6502 with a value. The immediate addressing mode means that the value needed will be stored along with the program - the byte after the LDY opcode - the CPU does not need to fetch it from another memory location.

4. Look up the opcode for LDX Immediate addressing and put it in the third box. Enter the hex value for the number 1 in the 4^{th} box. The "OLOOP" is a label that means "outer loop", which is only for human reference, so we can refer to this LDX instruction later. The X and Y registers are called Index registers because they are typically used to calculate offsets from a memory address, as we will see next.

5. The label "ILOOP" means Inner Loop. For each of the 20 lines (the Outer Loop), we need to print all the characters of our message (the Inner Loop). The instruction LDA $340,X means to load the A register (also known as the Accumulator) with the value from memory address calculated as $340 plus the value of the X register. Look up the opcode for the Indexed Absolute addressing mode that uses the X register. Since X currently equal 1, it will load the A register with the memory value from address $341, which contains the Ascii value of the 'H' character in the word "HELLO". In order to tell the computer the address $340, it needs to be stored in 2 bytes, with the LOW byte of the address first, followed by the HIGH byte. For $340, the LOW byte is $40 and the HIGH byte is $03. This needs to be done because the value $340 is greater than the largest number that can fit in 1 byte: $FF or 255 decimal.

6. The JSR $FFEF instruction calls a program written by Steve Wozniak in 1977, which is hard wired in the ROM of the Apple 1. When you call that program (known as a subroutine) the Ascii character in the Accumulator gets printed to the monitor (or TV, used by most Apple 1 users in 1977). After you lookup the JSR opcode (which means Jump to SubRoutine), remember to put the LOW byte of the $FFEF address first, followed by the HIGH byte.

7. Look up the INX (Increment X) opcode. It is a 1 byte instruction.

8. Look up the CPX opcode for the Absolute addressing mode. This instruction means "ComPare X to something". Then encode the address $0340, LOW byte followed by HIGH byte. When the comparison is done, flags are set in the 6502 Processor Status register to indicate the result of the compare, as follows:

    a. If X is less than the value, the C (carry) flag is clear (0).
    b. If X is greater than or equal to the value, the C (carry flag is set (1)).
    c. If X exactly equals the value, the Z (zero) flag is set (1).
    d. If X does not exactly equal the value, the Z (zero) flag is clear (0).

9. The instruction BCC ILOOP is a Branching instruction, which means that if the specified condition is met, the next instruction executed by the CPU will be the instruction at the specified label. If the condition is not met, the program flow will simply continue with the next instruction. BCC means Branch if Carry is Clear, so if the X register is less than the value in memory address $340, this instruction will branch to the instruction at the ILOOP label. However, the 6502 doesn't know what ILOOP is. Instead it needs to know how many bytes to go forwards or backwards from the current instruction. So count the number of boxes starting at the 2nd byte of the BCC ILOOP instruction, and going back until you reach the 1st byte of the LDA $340 instruction. Then look up that negative number in the following table to get the value for the 2nd byte of the BCC ILOOP instruction. This addressing mode is called Relative Addressing.

Decimal	-1	-2	-3	-4	-5	-6	-7	-8	-9
Hex	FF	FE	FD	FC	FB	FA	F9	F8	F7

Decimal	-10	-11	-12	-13	-14	-15	-16	-17	-18
Hex	F6	F5	F4	F3	F2	F1	F0	EF	EE

Decimal	-19	-20	-21	-22	-23	-24	-25	-26	-27
Hex	ED	EC	EB	EA	E9	E8	E7	E6	E5

10. The instruction BEQ ILOOP means Branch if EQual. It checks the Z register. If it equals 1, the branch is taken. Otherwise, (if Z=0) it continues with the next instruction. After you look up the opcode for BEQ, use the table above to find the offset. Note that the combination of the BCC and BEQ instructions are needed to check if X is less than or equal to the value in $340.

11. Look up the opcode for the LDA immediate instruction. The value $8D is the hi-bit Ascii value for a Carriage Return. Apple computers use Ascii values between 128 and 255. The Return key on the Apple is the same as Control M. M is the 13th letter of the alphabet, which has the hex value $D. Therefore, the Apple value for the carriage return is $8D which equals 128+13 or 141.

12. The same JSR instruction we have seen before is being called to output the Carriage Return character. This will return the cursor to the first character position on the next line, where we will continue printing the next line of text.

13. Look up the opcode for DEY - it means DEcrement the Y register, which subtracts 1 from Y. Then for the BNE OLOOP instruction, count the boxes and lookup the offset to get back to the OLOOP instruction.

14. Look up the opcode for RTS - it means ReTurn from Subroutine. This will return to the caller of our program (which may be another machine language program, the Apple monitor, or a Basic program that called our subroutine).

15. When you have filled in all the opcodes, and have looked up the Ascii hex values for your name, you are ready to type your program into the Replica 1 Plus and run it. See the Beginner's Instructions for details about how to type in hex codes for a program. When you have entered and verified your program, you can run it using the command: **300 R**

If your program runs successfully, Congratulations! Take a photo with your program and its output.

If not, don't be discouraged. Most machine language programs do NOT work on the first try. Now you will learn how to troubleshoot your program.

a. Type the following command to display the hex bytes for your program: **300.340**

b. Verify that you typed all the hex codes correctly for your program.

c. If you are sure that you typed in all the codes correctly, run a disassembler program so you can see how the computer interpreted your hex codes. If the instructions shown do not match exactly with the listing on the first page, determine which bytes were incorrectly entered. Make a list of the addresses and values that were incorrect. You will then patch them with the correct values and try again.

d. If you get stuck, ask a friend to take a look at your program. Another set of eyes often finds bugs.

e. Sometimes when you are writing your own programs and you get stuck it may make sense to start over. There are also tools that can help you step through your program as it runs and examine memory locations and register values.

6502 OPCODES (Operation Codes)

ADC	Imm	69	Add with Carry
ADC	ZP	65	
ADC	ZP, X	75	
ADC	Abs	6D	
ADC	Abs, X	7D	
ADC	Abs, Y	79	01 + 11 = 100
ADC	(Indir, X)	61	
ADC	(Indir), Y	71	
AND	Imm	29	
AND	ZP	25	00→0
AND	ZP, X	35	01→0
AND	Abs	2D	10→0
AND	Abs, X	3D	11→1
AND	Abs, Y	39	
AND	(Indir, X)	21	
AND	(Indir), Y	31	
ASL	Acc	0A	Shift Left
ASL	ZP	6	
ASL	ZP, X	16	
ASL	Abs	0E	
ASL	Abs, X	1E	
BCC		90	BRANCHES
BCS		B0	
BEQ		F0	
BIT	ZP	24	
BIT	Abs	2C	
BMI		30	
BNE		D0	
BPL		10	
BRK		0	
BVC		50	Processor Status Flags (N V B D I Z C)
BVS		70	
CLC		18	CLEAR
CLD		D8	
CLI		58	
CLV		B8	
CMP	Imm	C9	Compare
CMP	ZP	C5	

CMP	ZP, X	D5	
CMP	Abs	CD	
CMP	Abs, X	DD	
CMP	Abs, Y	D9	
CMP	(Indir, X)	C1	
CMP	(Indir), Y	D1	
CPX	Imm	E0	
CPX	ZP	E4	
CPX	Abs	EC	
CPY	Imm	C0	
CPY	ZP	C4	
CPY	Abs	CC	
DEC	ZP	C6	-1
DEC	ZP, X	D6	
DEC	Abs	CE	
DEC	Abs, X	DE	
DEX		CA	
DEY		88	
EOR	Imm	49	
EOR	ZP	45	00→0, 01→1
EOR	ZP, X	55	10→1, 11→0
EOR	Abs	4D	
EOR	Abs, X	5D	
EOR	Abs, Y	59	
EOR	(Indir, X)	41	
EOR	(Indir), Y	51	
INC	ZP	E6	+1
INC	ZP, X	F6	
INC	Abs	EE	
INC	Abs, X	FE	
INX		E8	
INY		C8	
JMP	Indir	6C	JUMP
JMP	Abs	4C	
JSR		20	
LDA	Imm	A9	
LDA	ZP	A5	
LDA	ZP, X	B5	

LDA	Abs	AD	
LDA	Abs, X	BD	
LDA	Abs, Y	B9	
LDA	(Indir, X)	A1	
LDA	(Indir), Y	B1	
LDX	ZP	A6	
LDX	ZP, Y	B6	
LDX	Abs	AE	
LDX	Abs, Y	BE	
LDX	Imm	A2	
LDY	Imm	A0	
LDY	ZP	A4	
LDY	ZP, X	B4	
LDY	Abs	AC	
LDY	Abs, X	BC	
LSR	Acc	4A	Shift RIGHT
LSR	ZP	46	
LSR	ZP, X	56	
LSR	Abs	4E	
LSR	Abs, X	5E	
NOP		EA	Do Nothing!
ORA	Imm	9	
ORA	ZP	5	00→0, 01→1
ORA	ZP, X	15	10→1, 11→1
ORA	Abs	0D	
ORA	Abs, X	1D	
ORA	Abs, Y	19	
ORA	(Indir, X)	1	
ORA	(Indir), Y	11	
PHA		48	PUSH ON STACK
PHP		8	
PLA		68	PULL FROM STACK
PLP		28	
ROL	Acc	2A	
ROL	ZP	26	
ROL	ZP, X	36	
ROL	Abs	2E	
ROL	Abs, X	3E	

ROR	Acc	6A	
ROR	ZP	66	
ROR	ZP, X	76	
ROR	Abs	6E	
ROR	Abs, X	7E	
RTI		40	Return from Interrupt
RTS		60	
SBC	Imm	E9	Subtract with Carry (Borrow)
SBC	ZP	E5	
SBC	ZP, X	F5	
SBC	Abs	ED	
SBC	Abs, X	FD	
SBC	Abs, Y	F9	
SBC	(Indir, X)	E1	
SBC	(Indir), Y	F1	
SEC		38	Set Flags
SED		F8	
SEI		78	
STA	ZP	85	
STA	ZP, X	95	
STA	Abs	8D	
STA	Abs, X	9D	
STA	Abs, Y	99	
STA	(Indir, X)	81	
STA	(Indir), Y	91	
STX	ZP	86	
STX	ZP, Y	96	
STX	Abs	8E	
STY	ZP	84	
STY	ZP, X	94	
STY	Abs	8C	
TAX		AA	Transfers
TAY		A8	
TSX		BA	
TXA		8A	
TXS		9A	
TYA		98	

Your Power Tools → Addressing Modes → Newton

Instructions — Opcodes

Apple uses High Bit Ascii

The leftmost bit = 1
So add 128 to decimal value

When coding your name:
4x → Cx
5x → Dx

Decimal - Binary - Octal - Hex – ASCII Conversion Chart

Add $80 to Hex value

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32 (160)	00100000	040	20 ($A0)	SPACE	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	"	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	'	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(	72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29	)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR (8D)	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[	123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	|
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D	]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

Notice 0–15
$0–$F Hex
0000–1111 Binary

Line Feed → 10
Carriage Return → 13

Control Characters