nesdoug

29. ASM part 4

Yet another 6502 ASM lesson.

Arrays

The way to access arrays in 6502 ASM is to use indexed addresses. The X (or Y) register is used as the indexer. As usual, X=0 will get the first byte of the array.

```
LDX #0         load X with value 0
LDA Array1, X  will load A from the address Array + X
STA foo        A = $3f, store A at foo
...



Array1:
.byte $3f, $4f, $5f, $6f
```

*Warning, if your array address is in the zero page, and your index would put the address in the next page, it won't fetch from the $100 page, but rather from zero-page.

This is a bug of zero-page indexing on the 6502 processor. If you absolutely must put an array half in the zero-page and half out (I don't know why you would), you can force the assembler to use an 'absolute address'…ie. a 16-bit address, like this…

LDA a:Array2, X ;this will correctly get the byte from the $100 page

Here's another array example using the Y register.

LDY #1
LDA Array1, Y

And, you can use STA the same way…to fill an array.
LDA #1
LDX #5
STA Array1, X store the value 1, at the address 'Array1' + 5

Loops

Loops are fairly easy…

for (X = 0;X < 50; X++)

```
    LDX #0
  Loop:
    ...some code...
    INX
    CPX #50   compare X to 50
    BNE Loop  not 50, branch back to Loop
```

It can also be done like this…

```
    LDX #50
  Loop:
    ...some code...
    DEX         X--, if result = 0, sets zero flag
    BNE Loop    if no zero flag, branch back to Loop
```

Bigger Loop, if you need a loop bigger than 256

```
    LDY #4
    LDX #0
  Loop:
    ...some code...
    DEX
    BNE Loop
    DEY
    BNE Loop  Will loop 1024 times
```

Way bigger than anyone will ever need Loop, just for fun…

```
    LDA #5
    STA counter
    LDY #0
    LDX #0
  Loop:
    ...some code...
    DEX
    BNE Loop
    DEY
    BNE Loop
    DEC counter
    BNE Loop  256*256*5 = 327680 times
```

Indirect Indexing

LDA (ZP_address,X)
LDA (ZP_address),Y
The first…(ZP_address, X)…I never use, and I don't like it, so I'm going to skip it altogether. Sorry. I've never seen any code that uses it.

The second…(ZP_address), Y…is very useful. It's the 6502 equivalent of a pointer. You store an address in the zero-page, and you can access the data at the address that it points to…or index from that address with the Y register.

pointer = 2 zero-page addresses reserved

```
  LDA #<SOME_ARRAY
  STA pointer
  LDA #>SOME_ARRAY
  STA pointer+1
  LDY #0
  LDA (pointer), y load a from address pointer is pointing to...SOME_ARRAY
    A = $5e
  LDY #1
  LDA (pointer), y load a from address pointer is pointing to plus Y...SOME_ARRAY +
  SOME_ARRAY:
  .byte $5e, $7f
```

Let's say you have multiple rooms in the game, and you want to load the graphics for room #3. So, you index to a list of addresses of each room's data, and store the address in the zero-page, and now you can indirect index from that address using the Y register as the indexer. In this example, pointer and pointer+1 are zero-page addresses.

```
  LDA room  room = 3
  ASL A     we multiply by 2, because each address is 2 bytes long
  TAX       transfer A to X
  LDA ADDRESSES, X load A with the low byte of the room address
  STA pointer  store A in the zero-page RAM
  LDA ADDRESSES+1, X load A with the high byte of the room address

  STA pointer+1  store A in the zero-page RAM
  LDY #0
LOOP:
  LDA (pointer), Y load A with the fist byte of the array Room3
  STA somewhere, Y Maybe we store this data to another array, for parsing later
  CMP #$ff         let's say, the data set is terminated with $ff
  BEQ EXIT_LOOP    if = $ff, leave this loop
  INY
  BNE LOOP         it will keep looping for 256 bytes,
                   when Y wraps around to zero
```

```
EXIT_LOOP:




ADDRESSES:
.word Room0, Room1, Room2, Room3
the assembler will replace these with the addresses of each label.




Room0:
...data for room0
Room1:
...data for room1
Room2:
...data for room2
Room3:
...data for room3
```

Multiple-condition If/then statements…some more examples.

if ((foo == 0)&&(bar < 20))…do code if both true

```
LDA foo   load A from address foo, sets a few flags, zero flag if = 0
BNE Skip_Ahead  skip the code if foo != 0
LDA bar   load A from address, bar
CMP #20   compare to value 20
BCS Skip_Ahead  skip the code if bar >= 20
...   some code here



Skip_Ahead:
```

if ((foo == 0) || (bar < 20))…do code if either true

```
  LDA foo
  BNE Check_Bar  skip if foo != 0, but also check bar
Do_Code:
  ...
  JMP Ahead
Check_Bar:
  LDA bar
  CMP #20
  BCC Do_Code  branch to Do_Code if bar < 20



Ahead:
```

March 13, 2016April 13, 2017 dougfraker