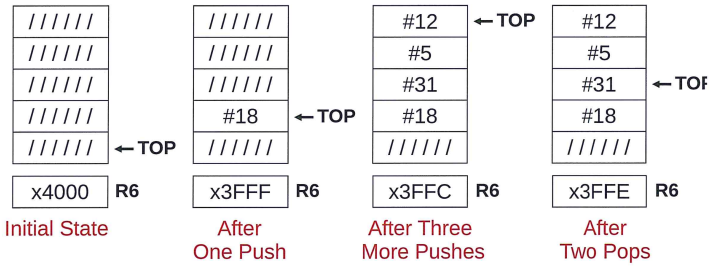
## 20. Examples
## Chapter 10

November 7, 2018

- Review
  - Hardware stack
  - Software implementation
  - PUSH and POP
- ASCII to Binary and Binary to ASCII conversion
- Examples

## A Software Implementation

**Data items don't move in memory,
just our idea about there the TOP of the stack is.**




**By convention, R6 holds the Top of Stack (TOS) pointer.**

## Pop with Underflow Detection

**If we try to pop too many items off the stack,
an underflow condition occurs.**

- Check for underflow by checking TOS before removing data.
- Return status code in R5 (0 for success, 1 for underflow)

```
POP   LD  R1, EMPTY  ; EMPTY = -x4000
      ADD R2, R6, R1 ; Compare stack pointer
      BRz FAIL       ; with x3FFF
      LDR R0, R6, #0
      ADD R6, R6, #1
      AND R5, R5, #0 ; SUCCESS: R5 = 0
      RET
FAIL  AND R5, R5, #0 ; FAIL: R5 = 1
      ADD R5, R5, #1
      RET
EMPTY .FILL xC000
```

## Push with Overflow Detection

**If we try to push too many items onto the stack,
an overflow condition occurs.**

- Check for underflow by checking TOS before adding data.
- Return status code in R5 (0 for success, 1 for overflow)

```
PUSH  LD  R1, MAX    ; MAX = -x3FFB
      ADD R2, R6, R1 ; Compare stack pointer
      BRz FAIL       ; with x3FFF
      ADD R6, R6, #-1
      STR R0, R6, #0
      AND R5, R5, #0 ; SUCCESS: R5 = 0
      RET
FAIL  AND R5, R5, #0 ; FAIL: R5 = 1
      ADD R5, R5, #1
      RET
MAX   .FILL xC005
```

## Data Type Conversion

**Keyboard input routines read ASCII characters, not binary values.**

**Similarly, output routines write ASCII.**

EXAMPLE:

```
IN              ; input from kbd.
ADD  R1, R0, #0 ; move to R1
IN              ; input from kbd
ADD  R0, R1, R0 ; add two inputs
OUT
HALT
```

USER inputs 2 and 3

Result ?? ~~A~~ e ??

[ASCII "2" (x32) + ASCII "3" (x33)

= x65 : ASCII "e"

## ASCII to Binary

**Useful to deal with mult-digit decimal numbers**

**Assume we've read three ASCII digits (e.g., "259") into a memory buffer.**

| | |
|---|---|
| x32 | '2' |
| x35 | '5' |
| x39 | '9' |

**How do we convert this to a number we can use?**

- **Convert first character to digit (subtract x30) and multiply by 100.**
- **Convert second character to digit and multiply by 10.**
- **Convert third character to digit.**
- **Add the three digits together.**

## Multiplication via a Lookup Table

**How can we multiply a number by 100?**

- **One approach:**
  **Add number to itself 100 times.**
- **Another approach:**
  **Add 100 to itself <number> times. (Better if number < 100.)**

**Since we have a small range of numbers (0-9), use number as an index into a lookup table.**

**Entry 0: 0 x 100 = 0**
**Entry 1: 1 x 100 = 100**
**Entry 2: 2 x 100 = 200**
**Entry 3: 3 x 100 = 300**
**etc.**

## Code for Lookup Table

```
; multiply R0 by 100, using lookup table
;
            LEA  R1, Lookup100  ; R1 = table base
            ADD  R1, R1, R0     ; add index (R0)
            LDR  R0, R1, #0     ; load from M[R1]
            ...
Lookup100   .FILL 0   ; entry 0
            .FILL 100 ; entry 1
            .FILL 200 ; entry 2
            .FILL 300 ; entry 3
            .FILL 400 ; entry 4
            .FILL 500 ; entry 5
            .FILL 600 ; entry 6
            .FILL 700 ; entry 7
            .FILL 800 ; entry 8
            .FILL 900 ; entry 9
```

## Complete Conversion Routine (1 of 3)

```
; Three-digit buffer at ASCIIBUF.
; R1 tells how many digits to convert.
; Put resulting decimal number in R0.
ASCIItoBinary  AND  R0, R0, #0  ; clear result
               ADD  R1, R1, #0  ; test # digits
               BRz  DoneAtoB    ; done if no digits
;
               LD   R3, NegZero ; R3 = -x30
               LEA  R2, ASCIIBUF
               ADD  R2, R2, R1
               ADD  R2, R2, #-1 ; points to ones digit
;
               LDR  R4, R2, #0  ; load digit
               ADD  R4, R4, R3  ; convert to number
               ADD  R0, R0, R4  ; add ones contrib
```

## Conversion Routine (2 of 3)

```
               ADD  R1, R1, #-1  ; one less digit
               BRz  DoneAtoB     ; done if zero
               ADD  R2, R2, #-1  ; points to tens digit
;
               LDR  R4, R2, #0   ; load digit
               ADD  R4, R4, R3   ; convert to number
               LEA  R5, Lookup10 ; multiply by 10
               ADD  R5, R5, R4
               LDR  R4, R5, #0
               ADD  R0, R0, R4   ; adds tens contrib
;
               ADD  R1, R1, #-1  ; one less digit
               BRz  DoneAtoB     ; done if zero
               ADD  R2, R2, #-1  ; points to hundreds
                                 ; digit
```

## Conversion Routine (3 of 3)

```
               LDR  R4, R2, #0   ; load digit
               ADD  R4, R4, R3   ; convert to number
               LEA  R5, Lookup100 ; multiply by 100
               ADD  R5, R5, R4
               LDR  R4, R5, #0
               ADD  R0, R0, R4   ; adds 100's contrib
;
DoneAtoB       RET
NegZero        .FILL xFFD0  ; -x30
ASCIIBUF       .BLKW 4
Lookup10       .FILL 0
               .FILL 10
               .FILL 20
...
Lookup100      .FILL 0
               .FILL 100
...
```

## Binary to ASCII Conversion

**Converting a 2's complement binary value to a three-digit decimal number**

- Resulting characters can be output using OUT

**Instead of multiplying, we need to divide by 100 to get hundreds digit.**

- Why wouldn't we use a lookup table for this problem?
- Subtract 100 repeatedly from number to divide.

**First, check whether number is negative.**

- Write sign character (+ or -) to buffer and make positive.

## Binary to ASCII Conversion Code (part 1 of 3)

```
; R0 is between -999 and +999.
; Put sign character in ASCIIBUF, followed by three
; ASCII digit characters.

BinaryToASCII  LEA R1, ASCIIBUF  ; pt to result string
               ADD R0, R0, #0    ; test sign of value
               BRn NegSign
               LD  R2, ASCIIplus ; store '+'
               STR R2, R1, #0
               BRnzp  Begin100
NegSign        LD  R2, ASCIIneg  ; store '-'
               STR R2, R1, #0
               NOT R0, R0        ; convert value to pos
               ADD R0, R0, #1
```

## Conversion (2 of 3)

```
Begin100       LD  R2, ASCIIoffset
               LD  R3, Neg100
Loop100        ADD R0, R0, R3
               BRn End100
               ADD R2, R2, #1  ; add one to digit
               BRnzp  Loop100
End100         STR R2, R1, #1  ; store ASCII 100's digit
               LD  R3, Pos100
               ADD R0, R0, R3  ; restore last subtract
;
               LD  R2, ASCIIoffset
               LD  R3, Neg10
Loop100        ADD R0, R0, R3
               BRn End10
               ADD R2, R2, #1  ; add one to digit
               BRnzp  Loop10
```

## Conversion Code (3 of 3)

```
End10           STR R2, R1, #2  ; store ASCII 10's digit
                ADD R0, R0, #10 ; restore last subtract
;
                LD  R2, ASCIIoffset
                ADD R2, R2, R0  ; convert one's digit
                STR R2, R1, #3  ; store one's digit
                RET

;
ASCIIplus    .FILL x2B   ; plus sign
ASCIIneg     .FILL x2D   ; neg sign
ASCIIoffset  .FILL x30   ; zero
Neg100       .FILL xFF9C ; -100
Pos100       .FILL #100
Neg10        .FILL xFFF6 ; -10
```