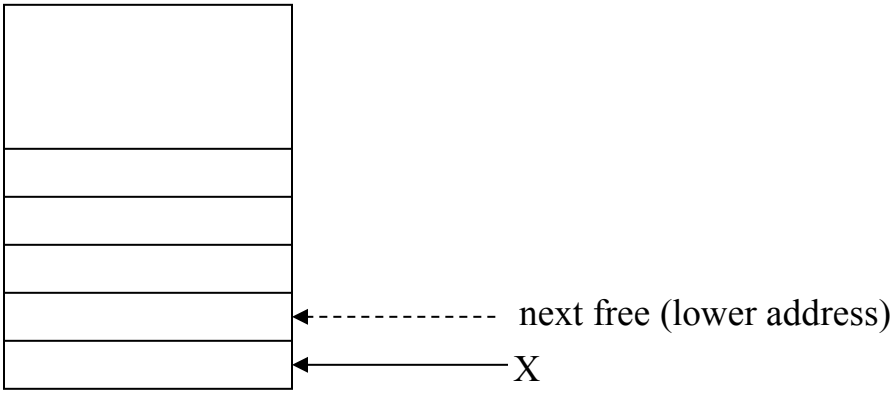
Digital and Embedded Systems
Professor Thomas Bräunl

Tutorial Optional B – Stack

1. Stack
A stack data structure can use either the ATMega's system stack (using the stack pointer SP) or a user-defined stack (using index registers X, Y or Z).

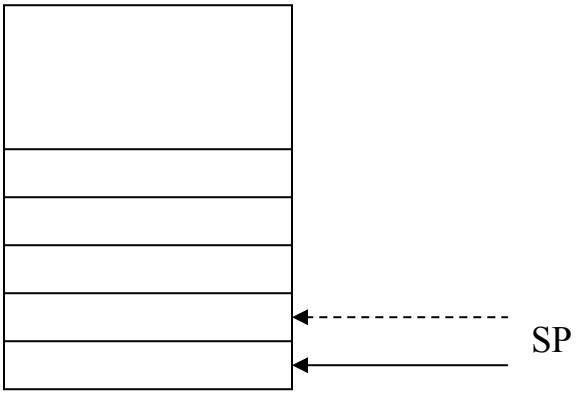
Following the Atmel convention of **post-decrement** for **push** and **pre-increment** for **pop**, implement your own subroutines *mypush* and *mypop*. Use X as a user stack-pointer and R16 as contents to be pushed/poped.



2. PUSH and POP
Execute the following Atmel code, draw the stack contents, and find out the register contents at the end. Assume SP is initialized with $04FF.

```
LDI  R16, $10
LDI  R17, $20

PUSH R16
PUSH R17
NOP
POP  R16
POP  R17
```



Contents of: R16 _____

R17 _____
SP _____

3. Machine Code and Status Register

Consider the machine program shown below. Fill in the instruction and parameters column by finding each instruction in the instruction set.

Address	Code	Instruction	Parameters
0	2700		
1	2711		
2	0000		
3	9100		
4	0400		
5	9110		
6	0401		
7	0F01		
8	9300		
9	0402		

Fill in the blanks in the program execution table below. Each line corresponds to a single instruction.

Enter the contents of the program counter, registers, memory locations and status register flags after each instruction is executed. Status flags N, Z, V and C stand for the negative, zero, overflow and carry flag, respectively.

Note:

- *Assume all flags are initially set to zero.*
- *The PC only increments by 1 or 2, depending whether an instruction takes 1 or 2 words (2 or 4 bytes).*

(before ex)

PC	R16	R17	$0400	$0401	$0402	V	N	Z	C
0	$4A	$7E	$55	$AC	$42	0	0	0	0
1			$55	$AC					
2			$55	$AC					
3			$55	$AC					
5			$55	$AC					
7			$55	$AC					
8			$55	$AC					
A			$55	$AC					