

Microcomputers I – CE 320

Mohammad Ghamari, Ph.D.

Electrical and Computer Engineering

Kettering University

Announcement

- You are going to have your last quiz next week on Friday, March 12
- Homework exercise 5 will be uploaded on BB this week.

Lecture 14: The Stack

Today's Topics

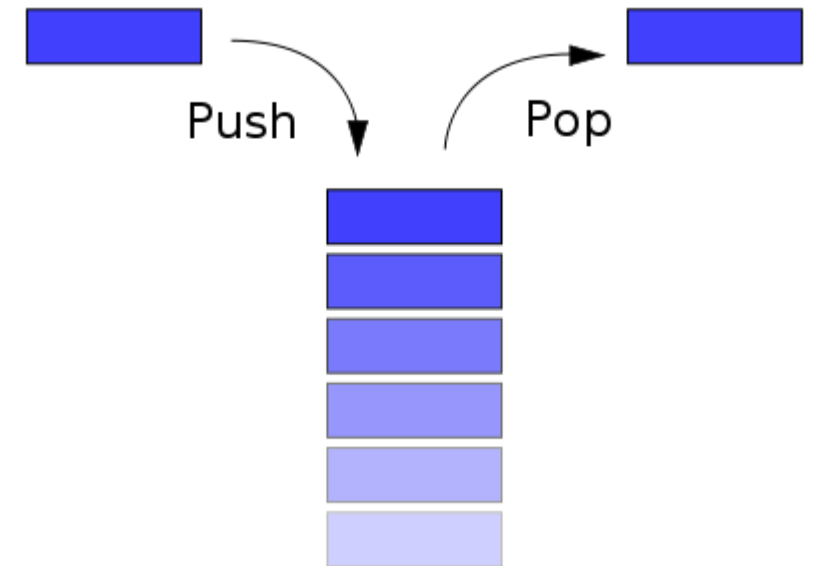
- Learn how HCS12 stack functions

What is the Stack?

Last In, First Out (LIFO)

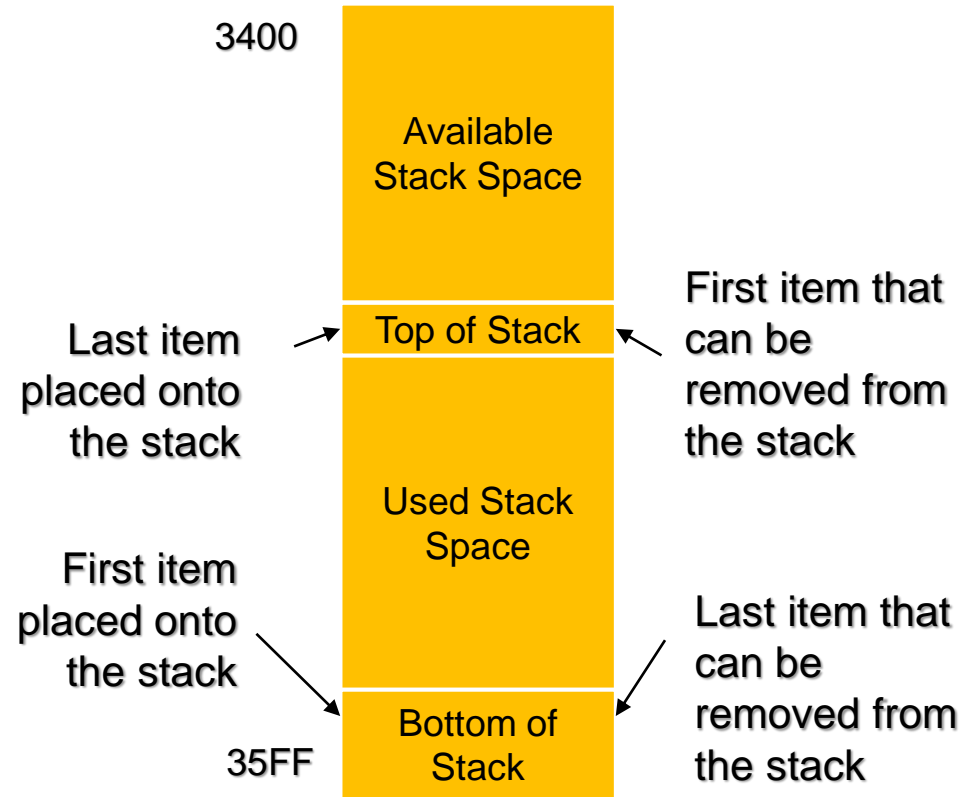
- The stack is a last-in, first-out (LIFO) data structure.
 - Elements can be accessed from only its top.
- Allows dynamic variable storage
 - Mainly used by microprocessors for calling subroutines.
- Two fundamental operations
 - Push: add to the top of the list.
 - Pop/Pull: removes an item from the top of the list.
- Think books stacked on a table
- c.f. Queue
 - First-in, first-out (FIFO) data structure

You will learn more about the **subroutines** later.



The Stack

Memory diagram of a stack



The Stack

- Physically, a stack can grow from a high address toward lower addresses or from a low address toward higher addresses.
- Here the HCS12 stack grows from a high address toward lower addresses and has a 16-bit stack pointer (SP) that points to the top byte of the stack.

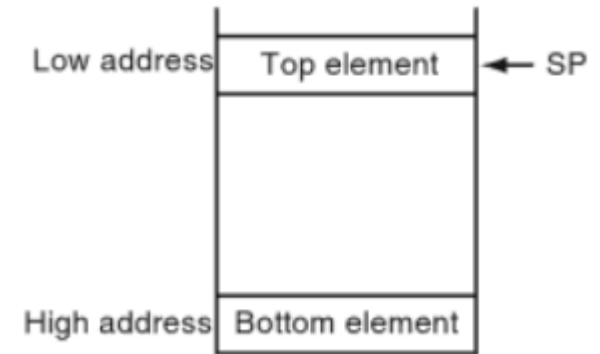


Diagram of the HCS12 stack

Important Concepts about Stacks

- A section of regular RAM (stack space) that all programs agree will be used for nothing else.
 - There is typically nothing in hardware that enforces this boundary.
- Much like variable-length array that uses a portion of the available stack space.
- Standard convention is that the stack grows towards lower addresses when data is added.
- The **stack pointer register (SP)** holds the address of the top byte and moves up (*decrements*) when items are added and moves down (*increments*) when data is removed.

Important Concepts about Stacks

- Depth:
 - The number of bytes stored on the stack
- Empty:
 - If the depth is 0, we say the stack is **empty**
- Underflow
 - Removing a byte from an empty stack
- Overflow
 - The depth is larger than the available stack space

Manipulating the Stack & Stack Pointer Register

Adding/Removing an item to a stack

- **PSHx**

- Push an item to the stack
- X: A, B, CCR (C), D, X, Y
- S register is first decremented by the number of bytes to be pushed, then the register value is copied into the newly generated “hole”.
- No affect on CCR bits

- **PULx**

- Pull or Pop an item from the stack
- X: A, B, CCR (C), D, X, Y
- One or two bytes at the top of the stack is/are copied into the specified register, then the S register is incremented.
- No affect on CCR (unless PULC)

- Again, there is no separate space for the stack. The stack is just a chunk of RAM. S register holds the current position.

Manipulating the Stack & Stack Pointer Register

- **LDS**
 - Loads the S (also referred as SP) register
 - Supports multiple addressing modes, but we will typically use only immediate addressing
 - Typically done once at the beginning of a program to “initialize” the stack, at least for this class.

Other Useful Stack Operations ...

- It is also useful at times to ***allocate space on the stack*** for variables without having the data to push onto the stack first.
- Also, we may want to ***remove a number of bytes*** without having to pull the unneeded data into registers with pull operations.
- This can be done using the ***Load Effective Address*** instruction.
- ***LEAS $\pm n$, SP***
 - Uses multiple addressing modes, but we will use indexed off of S almost exclusively.
 - With negative offsets, creates (allocates) space on the top of the stack.
 - With positive offsets, removes (de-allocates) data from the top of the stack.

Where is the Runtime Stack ?

- SP = address of the top element
- Before any PSH/PUL instruction, SP must be initialized.
 - LDS #\$3DFF
- Stack is any RAM area in main memory
- Who initializes ?
 - Simulator: Your program must use \$3DFF
 - NoICE: Auto-init's to \$3DFF



Example 1

Q: Assuming that we have the following instruction sequence to be executed by the HCS12, what would be the contents of the stack after the execution of these instructions?

1: LDS #\$3600
2: LDAA #\$AA
3: LDAB #\$BB
4: PSHA
5: PSHB
6: PULB
7: PULA

After Line 1		After Line 4		After Line 5		After Line 6		After Line 7	
35FD	<input type="text"/>	35FD	<input type="text"/>	35FD	<input type="text"/>	35FD	<input type="text"/>	35FD	<input type="text"/>
35FE	<input type="text"/>	35FE	<input type="text"/>	35FE	<input type="text"/>	35FE	<input type="text"/>	35FE	<input type="text"/>
35FF	<input type="text"/>	35FF	<input type="text"/>	35FF	<input type="text"/>	35FF	<input type="text"/>	35FF	<input type="text"/>
3600	<input type="text"/>	3600	<input type="text"/>	3600	<input type="text"/>	3600	<input type="text"/>	3600	<input type="text"/>
SP	<input type="text"/>	SP	<input type="text"/>	SP	<input type="text"/>	SP	<input type="text"/>	SP	<input type="text"/>

Example 1

Q: Assuming that we have the following instruction sequence to be executed by the HCS12, what would be the contents of the stack after the execution of these instructions?

1: LDS #\$3600
2: LDAA #\$AA
3: LDAB #\$BB
4: PSHA
5: PSHB
6: PULB
7: PULA

	After Line 1	After Line 4	After Line 5	After Line 6	After Line 7
35FD	XX	XX	XX	XX	XX
35FE	XX	XX	BB	XX	XX
35FF	XX	AA	AA	AA	XX
3600	XX	XX	XX	XX	XX
SP	3600	35FF	35FE	35FF	3600

Example 2

Q: Assuming that we have the following instruction sequence to be executed by the HCS12, what would be the contents of the stack after the execution of these instructions?

```
1:  LDS    #$3600
2:  LDAA   #$AA
3:  LDAB   #$BB
4:  PSHD
5:  LDD    #$CCDD
6:  LEAS   2,SP
```

After Line 1		After Line 4		After Line 5		After Line 6	
35FD	<input type="text"/>	35FD	<input type="text"/>	35FD	<input type="text"/>	35FD	<input type="text"/>
35FE	<input type="text"/>	35FE	<input type="text"/>	35FE	<input type="text"/>	35FE	<input type="text"/>
35FF	<input type="text"/>	35FF	<input type="text"/>	35FF	<input type="text"/>	35FF	<input type="text"/>
3600	<input type="text"/>	3600	<input type="text"/>	3600	<input type="text"/>	3600	<input type="text"/>
SP	<input type="text"/>	SP	<input type="text"/>	SP	<input type="text"/>	SP	<input type="text"/>

Example 2

Q: Assuming that we have the following instruction sequence to be executed by the HCS12, what would be the contents of the stack after the execution of these instructions?

```
1:  LDS    #$3600
2:  LDAA   #$AA
3:  LDAB   #$BB
4:  PSHD
5:  LDD    #$CCDD
6:  LEAS   2,SP
```

	After Line 1	After Line 4	After Line 5	After Line 6
35FD	XX	XX	XX	XX
35FE	XX	AA	AA	XX
35FF	XX	BB	BB	XX
3600	XX	XX	XX	XX
SP	3600	35FE	35FE	3600

Questions?

Wrap-up

What we've learned

- Stack
- PUSx, PULx, LDS, LEAS

What to Come

- Subroutines