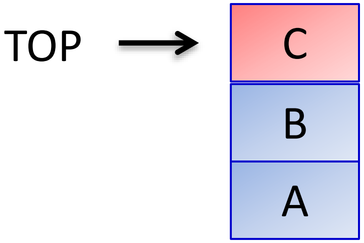
Background - Stacks

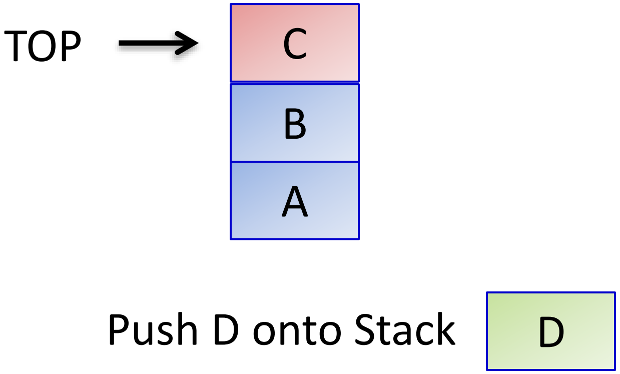
- A stack is an ordered array of data items that can only be accessed at one end of the array, called the TOP
  - A PUSH operation adds a new data item to the top of the stack
  - A POP operation removes the top item from the stack



Initial State of Stack

Background - Stacks

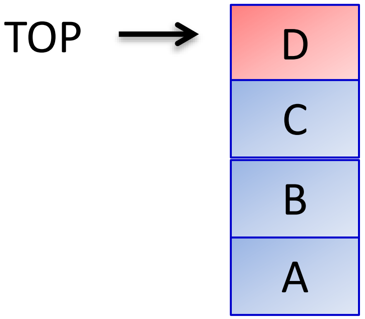
- A stack is an ordered list of data items that can only be accessed at one end of the list, called the TOP
  - A PUSH operation adds a new data item to the top of the stack
  - A POP operation removes the top item from the stack



Push D onto Stack

Background - Stacks

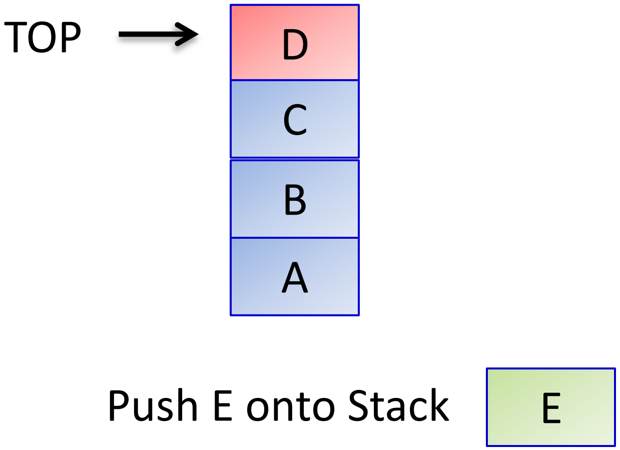
- A stack is an ordered list of data items that can only be accessed at one end of the list, called the TOP
  - A PUSH operation adds a new data item to the top of the stack
  - A POP operation removes the top item from the stack



New State of Stack

Background - Stacks

- A stack is an ordered list of data items that can only be accessed at one end of the list, called the TOP
  - A PUSH operation adds a new data item to the top of the stack
  - A POP operation removes the top item from the stack



TOP

D

C

B

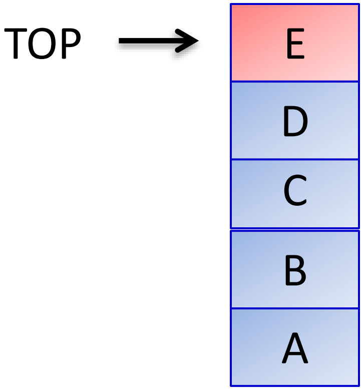
A

Push E onto Stack

E

Background - Stacks

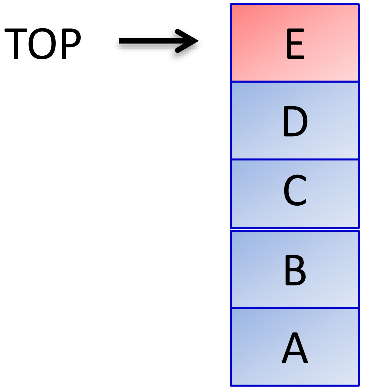
- A stack is an ordered list of data items that can only be accessed at one end of the list, called the TOP
  - A PUSH operation adds a new data item to the top of the stack
  - A POP operation removes the top item from the stack



New State of Stack

Background - Stacks

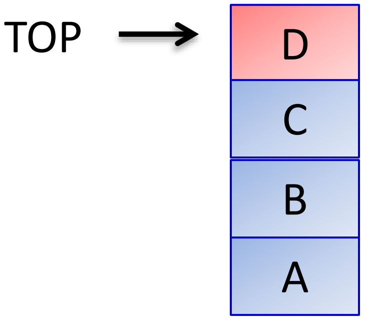
- A stack is an ordered list of data items that can only be accessed at one end of the list, called the TOP
  - A PUSH operation adds a new data item to the top of the stack
  - A POP operation removes the top item from the stack



Pop Stack

Background - Stacks

- A stack is an ordered list of data items that can only be accessed at one end of the list, called the TOP
  - A PUSH operation adds a new data item to the top of the stack
  - A POP operation removes the top item from the stack



New State of Stack

Stack-Based Languages

- Stack-based languages, like C, make use of stacks for the following purposes
  - Call-return (pointer) mechanism
  - Arguments
    - Pass-by-value
    - Pass-by-reference
  - Local variables
  - Nested function calls
  - Recursive function calls
  - Re-entrant Code

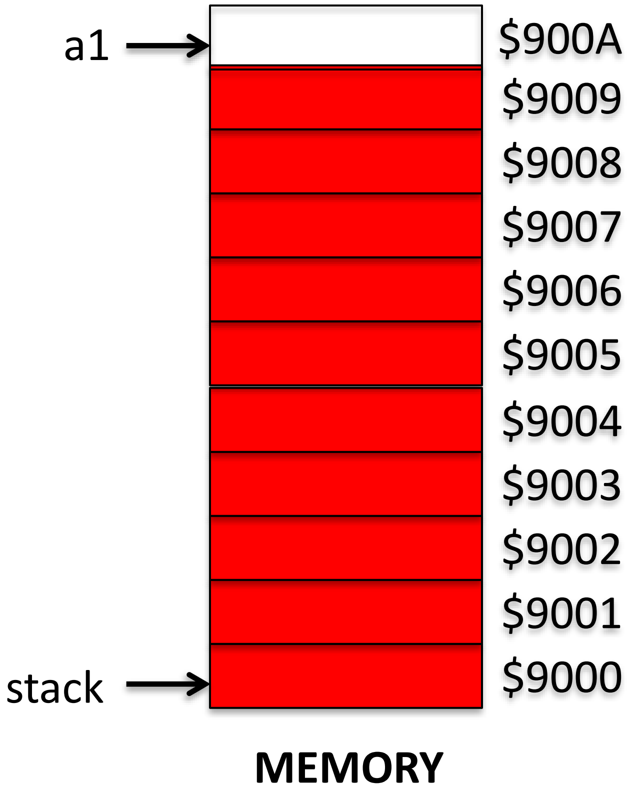
68000 Stack Conventions

- User Stacks on the 68000
  - grow from the “top” of memory towards the “bottom” of memory
  - employ an address register as a stack pointer to the top of the stack
    - System/User (runtime) stacks
      - two dedicated stack pointers: USP and SSP
      - both implemented using A7
    - Other stacks
      - 7 general-purpose stack pointers: A0-A6
  - the 68000 does not provide push and pop instructions
    - Push: **`MOVE <ea>,-(An)`**
    - Pop: **`MOVE (An)+,<ea>`**

Creating a (Private) User Stack

- Create a stack large enough to hold 10 bytes and use A1 as the stack pointer

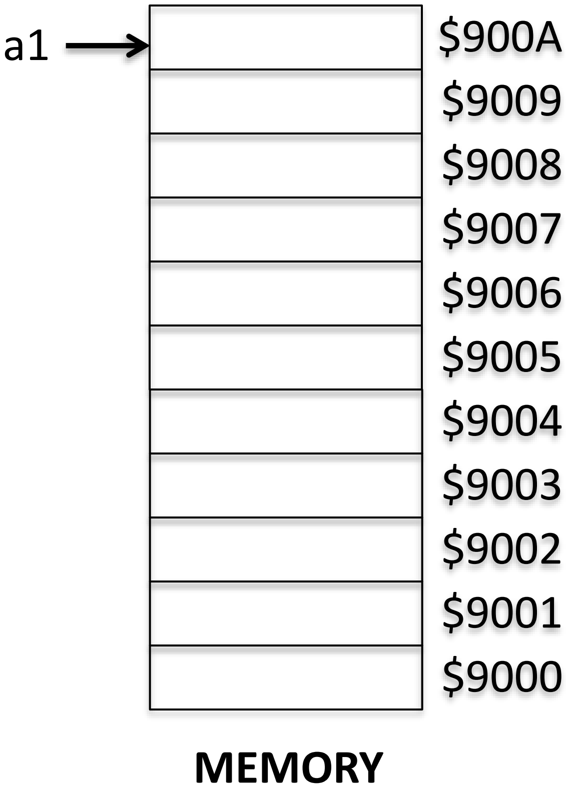
```
        org     $9000
stack   ds.b    10
        lea     stack+10, a1
```



Examples of Push and Pop Operations

d0	12	34	56	78
d1	00	00	9A	BC
d2	00	00	00	DE

```
move.l d0,-(a1)    ;push long word
move.w d1,-(a1)    ;push word
move.b d2,-(a1)    ;push byte
```



a1

$900A

$9009

$9008

$9007

$9006

$9005

$9004

$9003

$9002

$9001

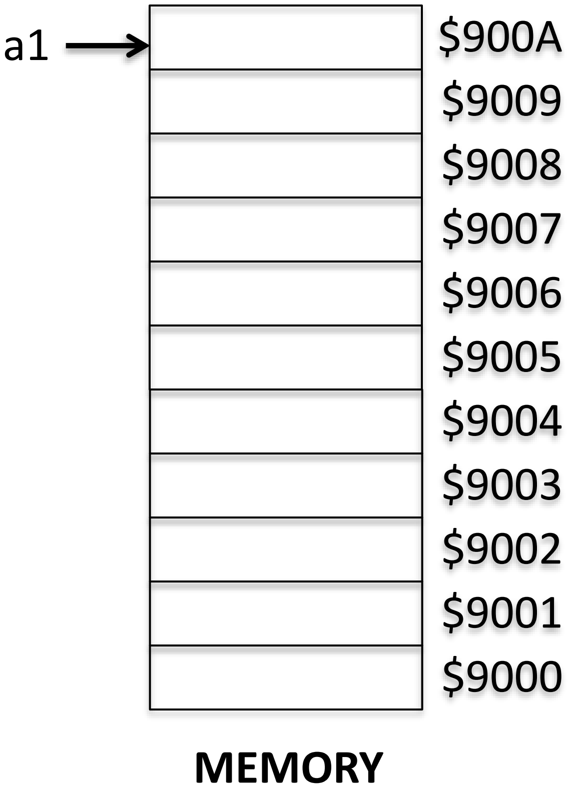
$9000

MEMORY

Examples of Push and Pop Operations

d0	12	34	56	78
d1	00	00	9A	BC
d2	00	00	00	DE

```
move.l d0,-(a1)    ;push long word
move.w d1,-(a1)    ;push word
move.b d2,-(a1)    ;push byte
```



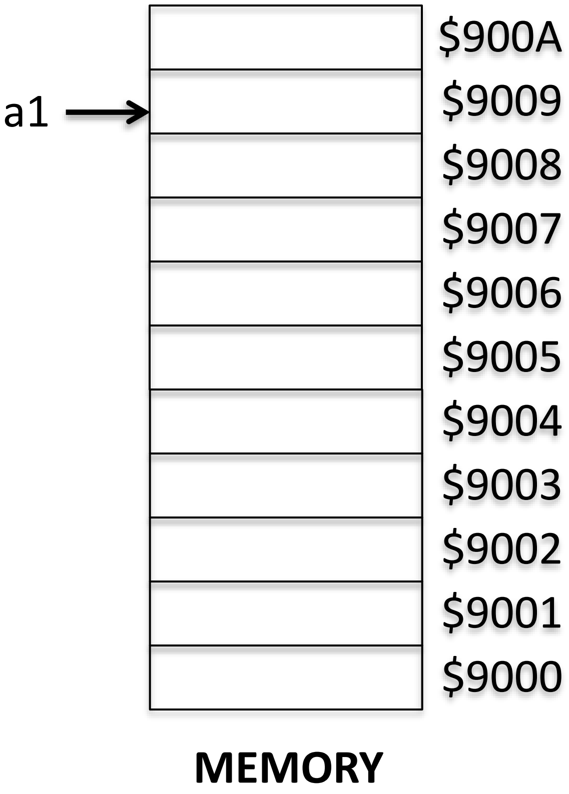
Examples of Push and Pop Operations

d0	12	34	56	78

d1	00	00	9A	BC

d2	00	00	00	DE

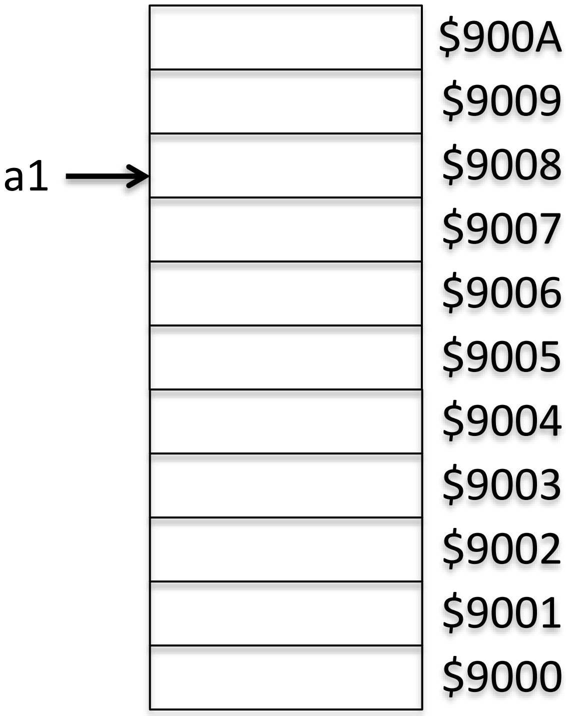
```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
```



Examples of Push and Pop Operations

d0	12	34	56	78
d1	00	00	9A	BC
d2	00	00	00	DE

```
move.l d0,-(a1)    ;push long word
move.w d1,-(a1)    ;push word
move.b d2,-(a1)    ;push byte
```



MEMORY

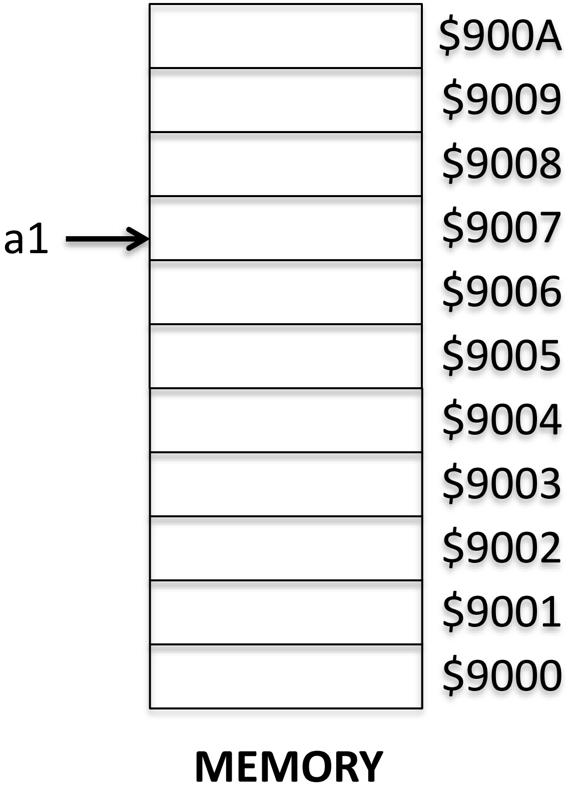
Examples of Push and Pop Operations

d0	12	34	56	78

d1	00	00	9A	BC

d2	00	00	00	DE

```
move.l d0,-(a1)    ;push long word
move.w d1,-(a1)    ;push word
move.b d2,-(a1)    ;push byte
```



a1

$900A

$9009

$9008

$9007

$9006

$9005

$9004

$9003

$9002

$9001

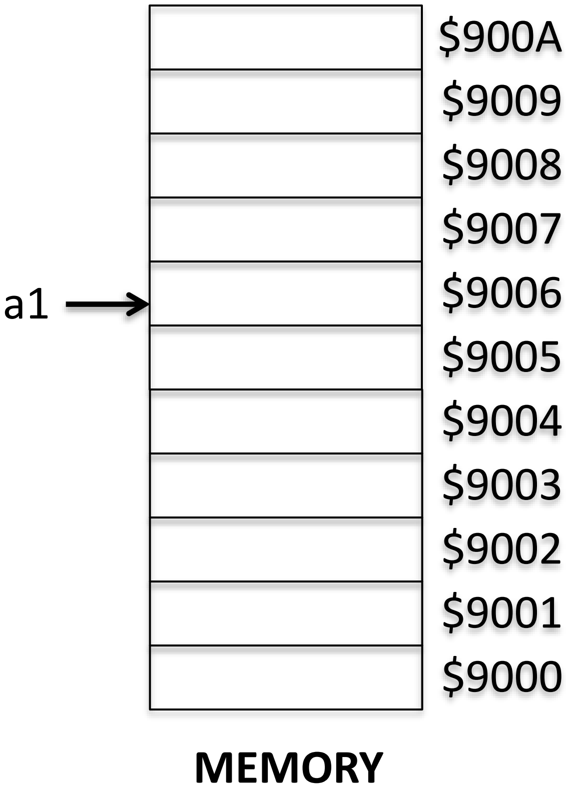
$9000

MEMORY

Examples of Push and Pop Operations

d0	12	34	56	78
d1	00	00	9A	BC
d2	00	00	00	DE

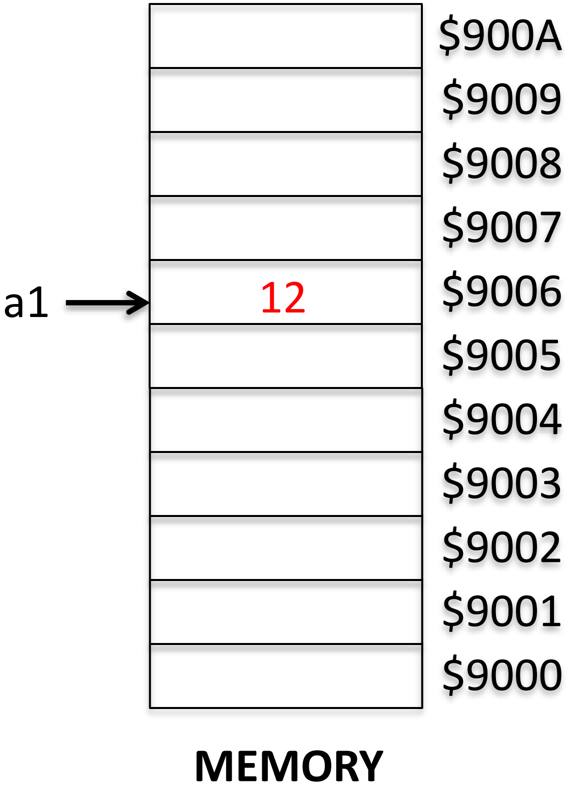
```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
```



Examples of Push and Pop Operations

d0	12	34	56	78
d1	00	00	9A	BC
d2	00	00	00	DE

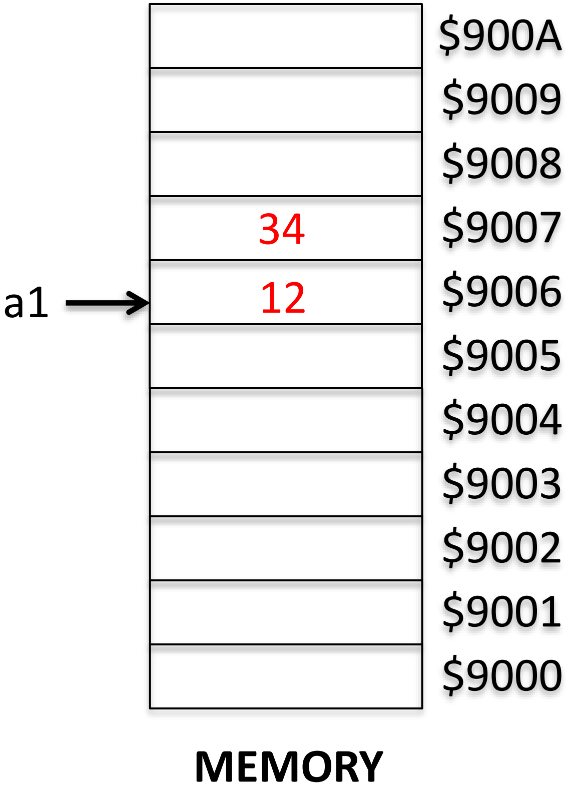
```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
```



Examples of Push and Pop Operations

d0	12	34	56	78
d1	00	00	9A	BC
d2	00	00	00	DE

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
```



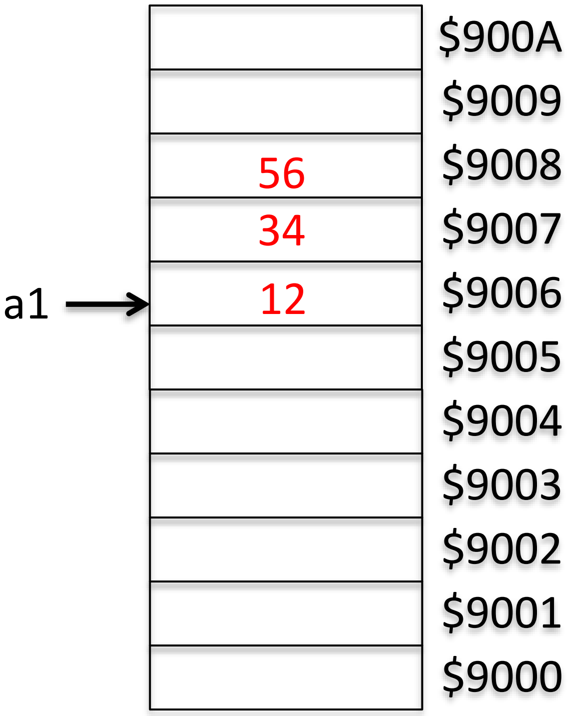
		$900A
		$9009
		$9008
	34	$9007
a1 →	12	$9006
		$9005
		$9004
		$9003
		$9002
		$9001
		$9000

MEMORY

Examples of Push and Pop Operations

d0	12	34	56	78
d1	00	00	9A	BC
d2	00	00	00	DE

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
```

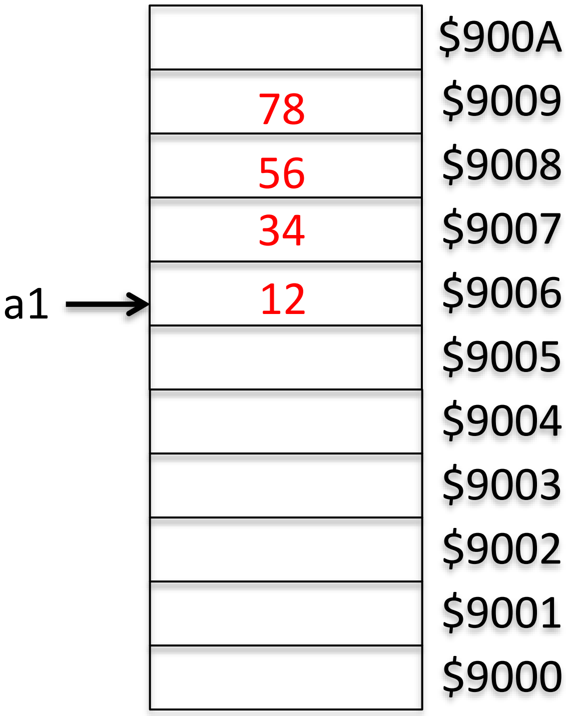


MEMORY

Examples of Push and Pop Operations

d0	12	34	56	78
d1	00	00	9A	BC
d2	00	00	00	DE

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
```



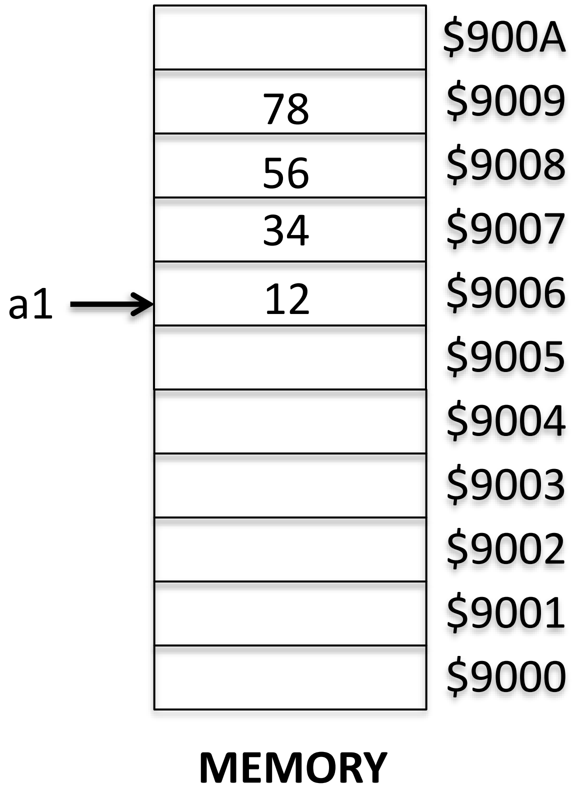
		$900A
	78	$9009
	56	$9008
	34	$9007
a1 →	12	$9006
		$9005
		$9004
		$9003
		$9002
		$9001
		$9000

MEMORY

Examples of Push and Pop Operations

d0	12	34	56	78
d1	00	00	9A	BC
d2	00	00	00	DE

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
```



		$900A
	78	$9009
	56	$9008
	34	$9007
a1 →	12	$9006
		$9005
		$9004
		$9003
		$9002
		$9001
		$9000

MEMORY

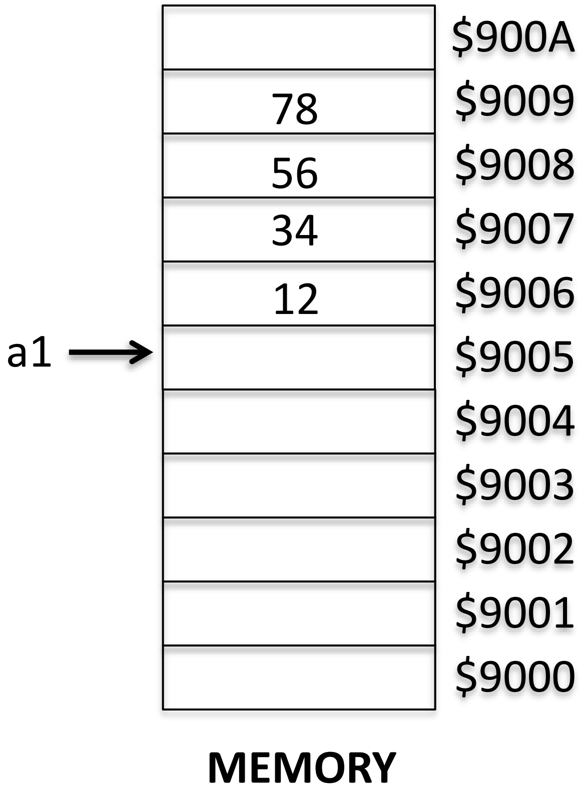
Examples of Push and Pop Operations

d0	12	34	56	78

d1	00	00	9A	BC

d2	00	00	00	DE

```
move.l d0,-(a1)    ;push long word
move.w d1,-(a1)    ;push word
move.b d2,-(a1)    ;push byte
```

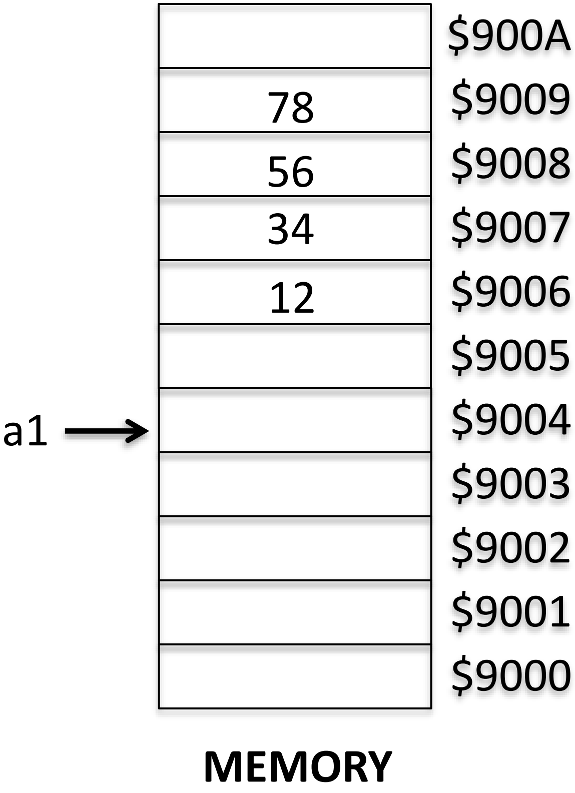


	$900A
78	$9009
56	$9008
34	$9007
12	$9006
	$9005
	$9004
	$9003
	$9002
	$9001
	$9000

Examples of Push and Pop Operations

d0	12	34	56	78
d1	00	00	9A	BC
d2	00	00	00	DE

```
move.l d0,-(a1)    ;push long word
move.w d1,-(a1)    ;push word
move.b d2,-(a1)    ;push byte
```



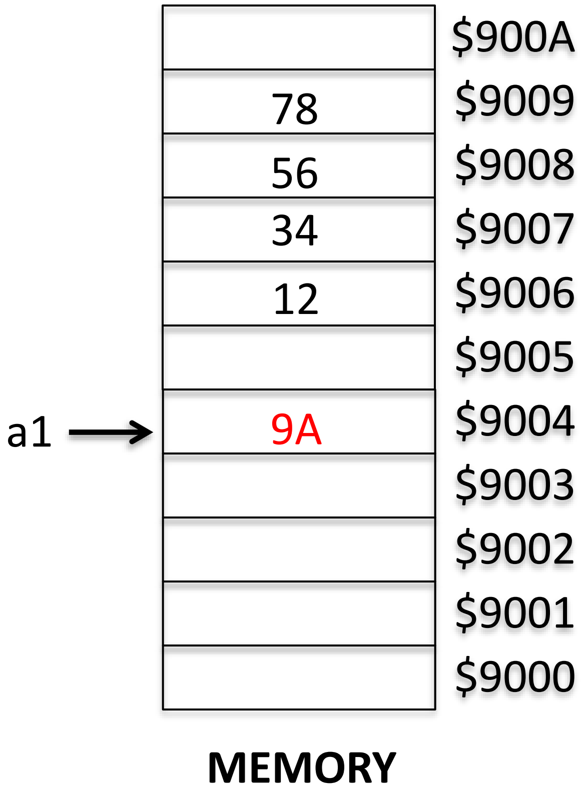
	$900A
78	$9009
56	$9008
34	$9007
12	$9006
	$9005
	$9004
	$9003
	$9002
	$9001
	$9000

MEMORY

Examples of Push and Pop Operations

d0	12	34	56	78
d1	00	00	9A	BC
d2	00	00	00	DE

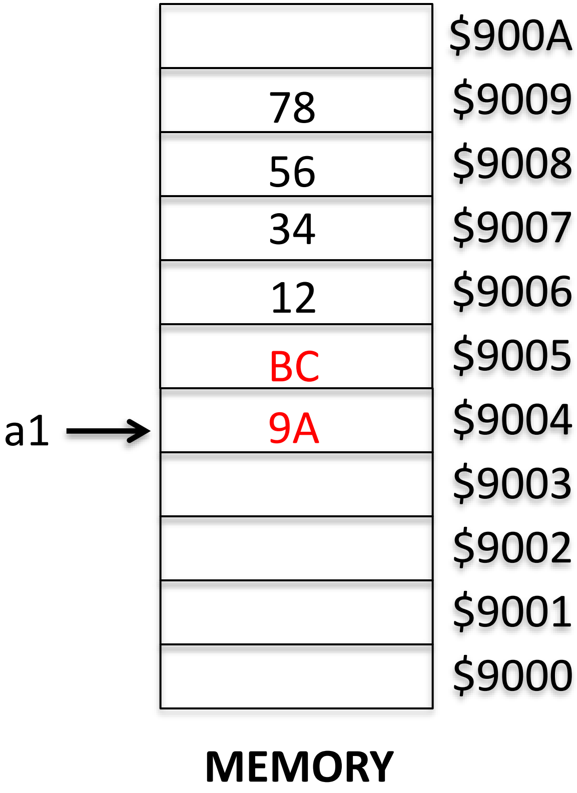
```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
```



Examples of Push and Pop Operations

d0	12	34	56	78
d1	00	00	9A	BC
d2	00	00	00	DE

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
```

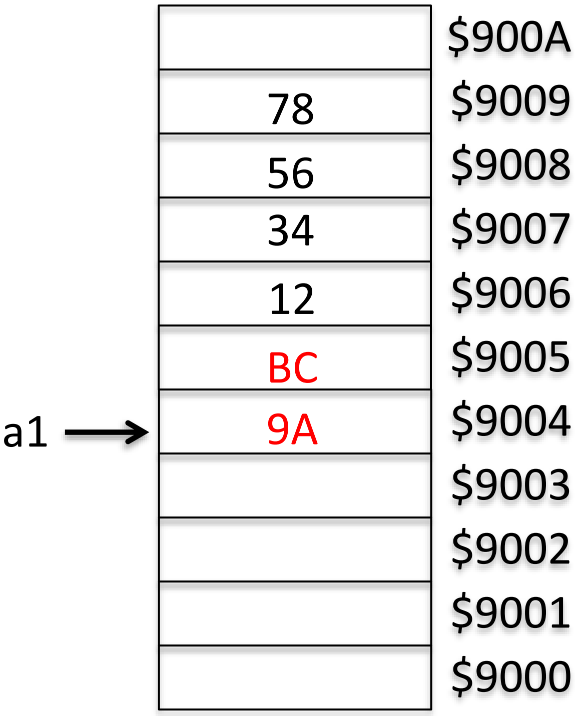


	$900A
78	$9009
56	$9008
34	$9007
12	$9006
BC	$9005
9A	$9004
	$9003
	$9002
	$9001
	$9000

Examples of Push and Pop Operations

d0	12	34	56	78
d1	00	00	9A	BC
d2	00	00	00	DE

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
```



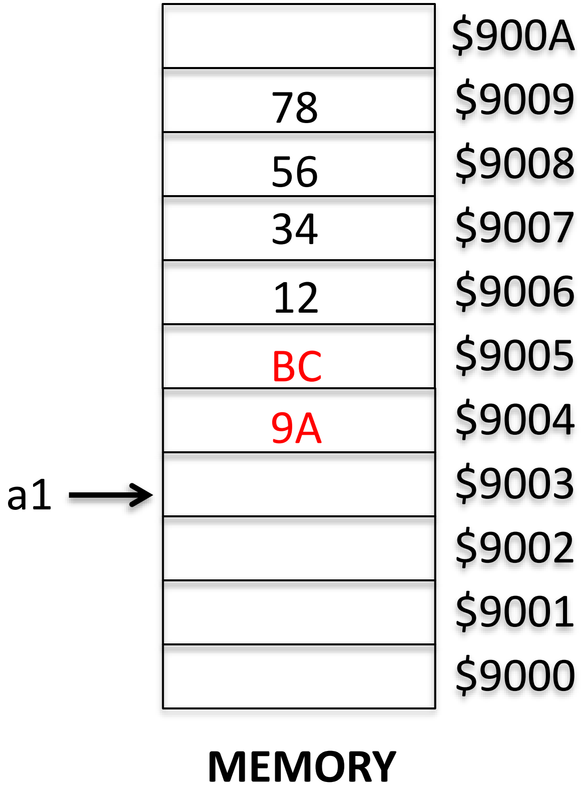
	$900A
78	$9009
56	$9008
34	$9007
12	$9006
BC	$9005
9A	$9004
	$9003
	$9002
	$9001
	$9000

MEMORY

Examples of Push and Pop Operations

d0	12	34	56	78
d1	00	00	9A	BC
d2	00	00	00	DE

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
```

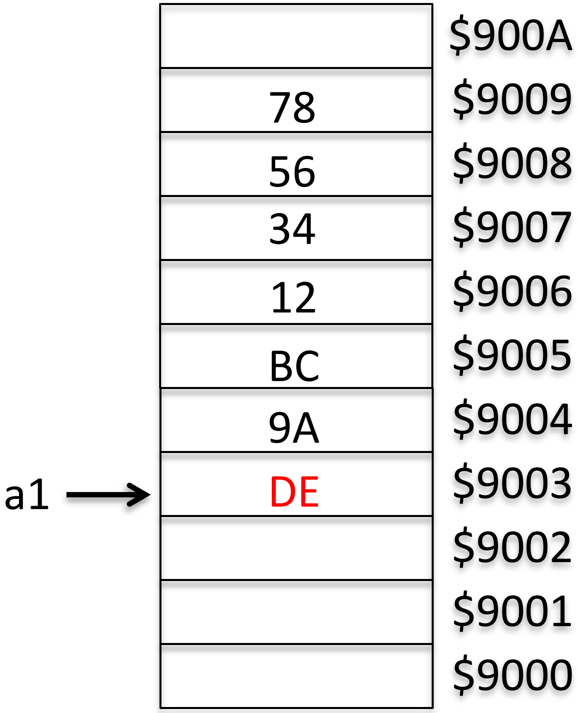


	$900A
78	$9009
56	$9008
34	$9007
12	$9006
BC	$9005
9A	$9004
	$9003
	$9002
	$9001
	$9000

Examples of Push and Pop Operations

d0	12	34	56	78
d1	00	00	9A	BC
d2	00	00	00	DE

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
```



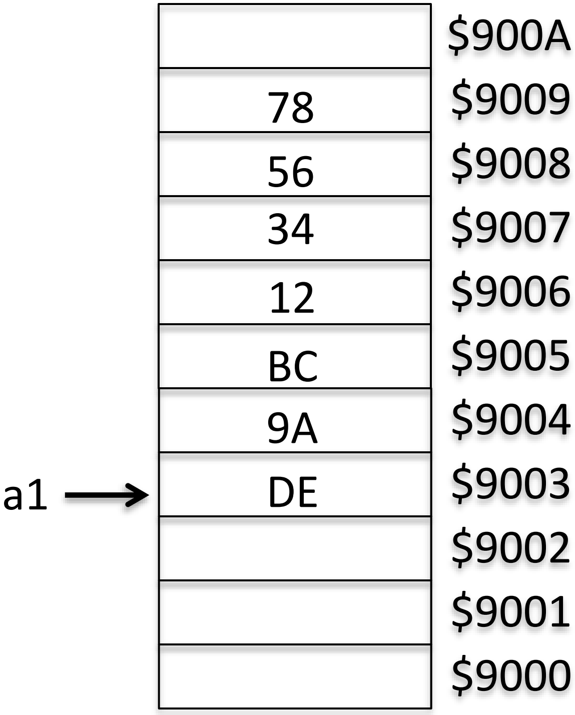
$900A	
$9009	78
$9008	56
$9007	34
$9006	12
$9005	BC
$9004	9A
$9003	DE
$9002	
$9001	
$9000	

MEMORY

Examples of Push and Pop Operations

d0	XX	XX	XX	XX
d1	XX	XX	XX	XX
d2	XX	XX	XX	XX

```
move.l d0,-(a1)    ;push long word
move.w d1,-(a1)    ;push word
move.b d2,-(a1)    ;push byte
                .
                .
                .
move.b (a1)+,d2    ;pop byte
move.w (a1)+,d1    ;pop word
move.l (a1)+,d0    ;pop long word
```



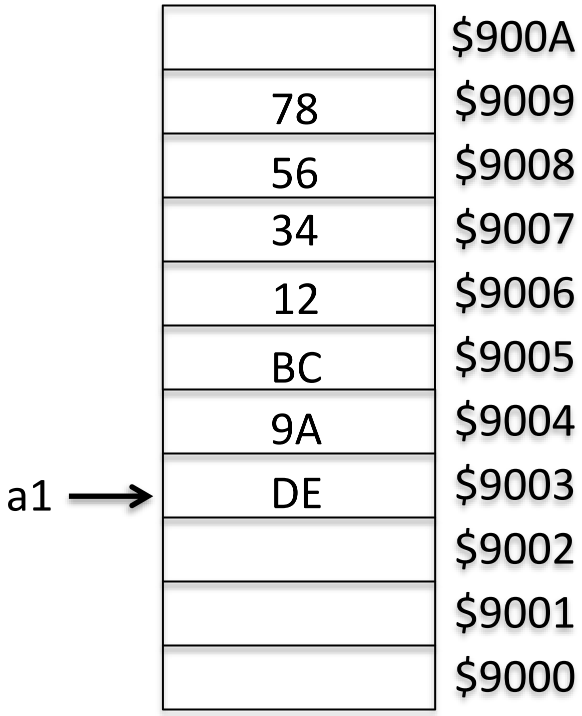
		$900A
	78	$9009
	56	$9008
	34	$9007
	12	$9006
	BC	$9005
	9A	$9004
a1 →	DE	$9003
		$9002
		$9001
		$9000

MEMORY

Examples of Push and Pop Operations

d0	XX	XX	XX	XX
d1	XX	XX	XX	XX
d2	XX	XX	XX	XX

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
                .
                .
                .
move.b (a1)+,d2     ;pop byte
move.w (a1)+,d1     ;pop word
move.l (a1)+,d0     ;pop long word
```



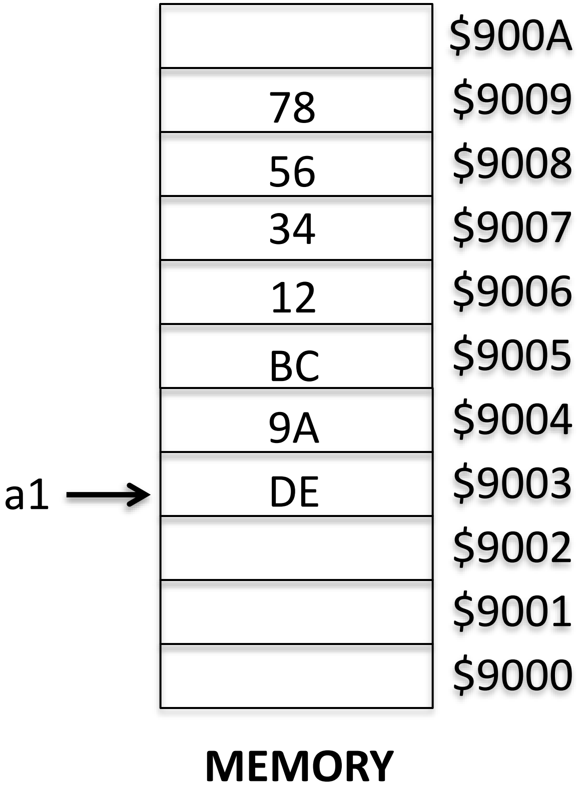
	$900A
78	$9009
56	$9008
34	$9007
12	$9006
BC	$9005
9A	$9004
DE	$9003
	$9002
	$9001
	$9000

MEMORY

Examples of Push and Pop Operations

d0	XX	XX	XX	XX
d1	XX	XX	XX	XX
d2	XX	XX	XX	DE

```
move.l d0,-(a1)    ;push long word
move.w d1,-(a1)    ;push word
move.b d2,-(a1)    ;push byte
                .
                .
                .
move.b (a1)+,d2    ;pop byte
move.w (a1)+,d1    ;pop word
move.l (a1)+,d0    ;pop long word
```



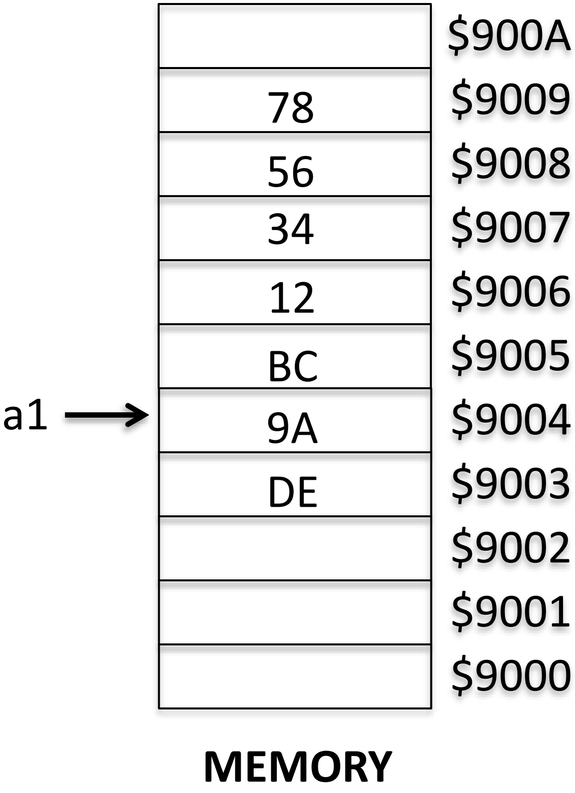
		$900A
	78	$9009
	56	$9008
	34	$9007
	12	$9006
	BC	$9005
	9A	$9004
a1 →	DE	$9003
		$9002
		$9001
		$9000

MEMORY

Examples of Push and Pop Operations

d0	XX	XX	XX	XX
d1	XX	XX	XX	XX
d2	XX	XX	XX	DE

```
move.l d0,-(a1)    ;push long word
move.w d1,-(a1)    ;push word
move.b d2,-(a1)    ;push byte
                .
                .
                .
move.b (a1)+,d2    ;pop byte
move.w (a1)+,d1    ;pop word
move.l (a1)+,d0    ;pop long word
```



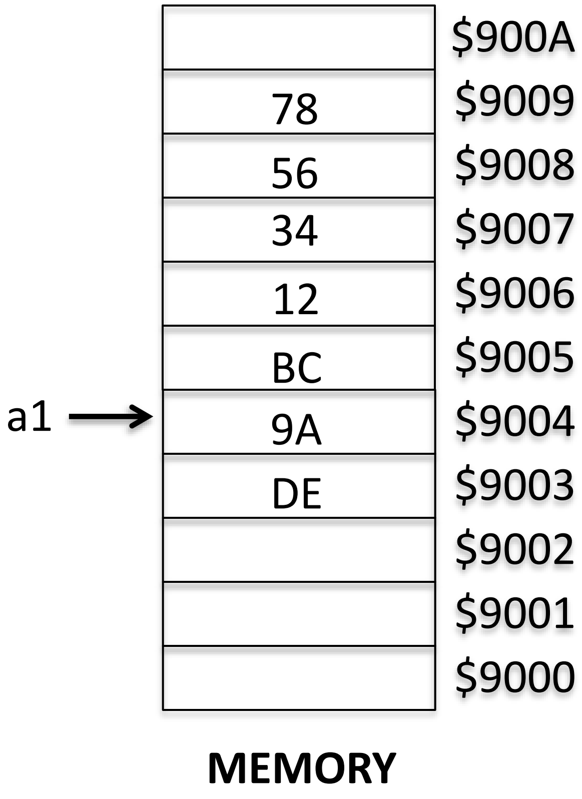
		$900A
	78	$9009
	56	$9008
	34	$9007
	12	$9006
	BC	$9005
a1 →	9A	$9004
	DE	$9003
		$9002
		$9001
		$9000

MEMORY

Examples of Push and Pop Operations

d0	XX	XX	XX	XX
d1	XX	XX	XX	XX
d2	XX	XX	XX	DE

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
                .
                .
                .
move.b (a1)+,d2     ;pop byte
move.w (a1)+,d1     ;pop word
move.l (a1)+,d0     ;pop long word
```



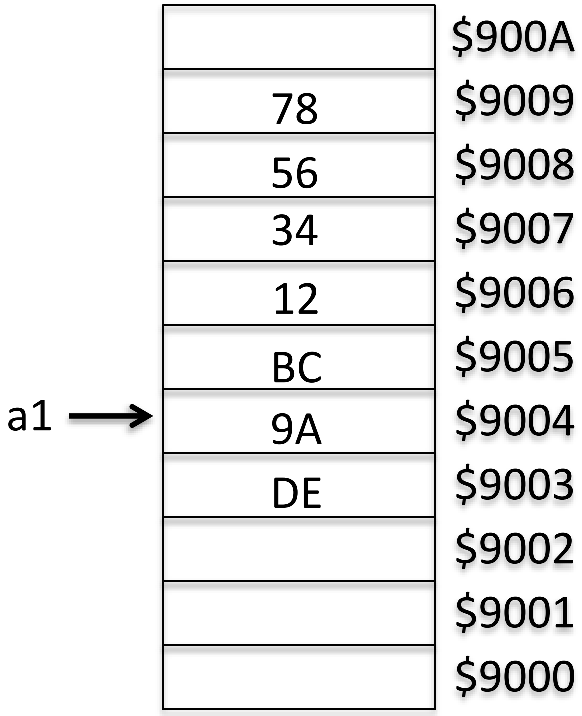
		$900A
	78	$9009
	56	$9008
	34	$9007
	12	$9006
	BC	$9005
a1 →	9A	$9004
	DE	$9003
		$9002
		$9001
		$9000

MEMORY

Examples of Push and Pop Operations

d0	XX	XX	XX	XX
d1	XX	XX	9A	BC
d2	XX	XX	XX	DE

```
move.l d0,-(a1)    ;push long word
move.w d1,-(a1)    ;push word
move.b d2,-(a1)    ;push byte
                .
                .
                .
move.b (a1)+,d2    ;pop byte
move.w (a1)+,d1    ;pop word
move.l (a1)+,d0    ;pop long word
```



		$900A
	78	$9009
	56	$9008
	34	$9007
	12	$9006
	BC	$9005
a1 →	9A	$9004
	DE	$9003
		$9002
		$9001
		$9000

MEMORY

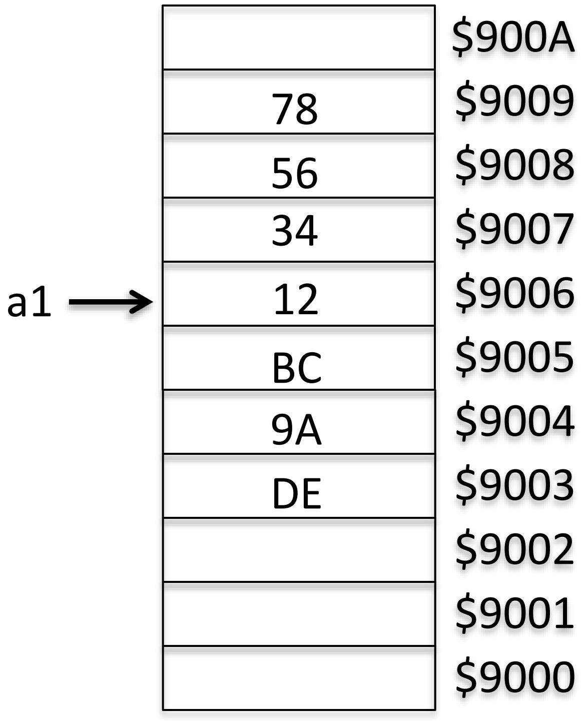
Examples of Push and Pop Operations

d0	XX	XX	XX	XX

d1	XX	XX	9A	BC

d2	XX	XX	XX	DE

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
                .
                .
                .
move.b (a1)+,d2     ;pop byte
move.w (a1)+,d1     ;pop word
move.l (a1)+,d0     ;pop long word
```



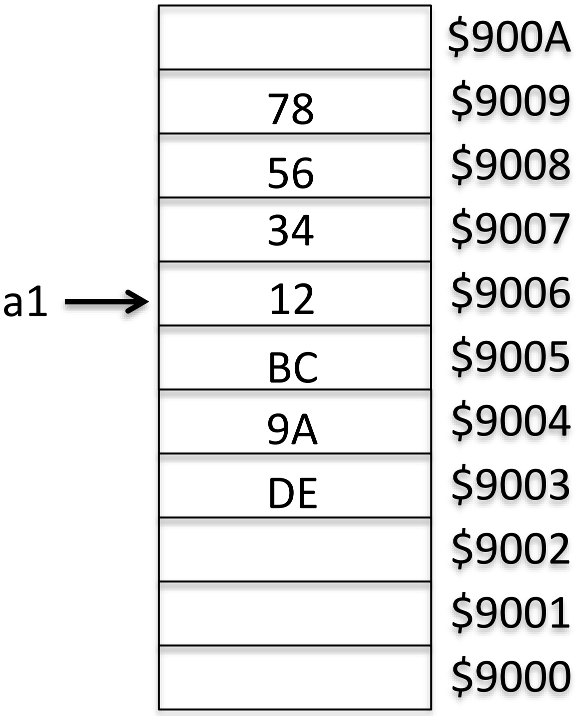
		$900A
	78	$9009
	56	$9008
	34	$9007
a1 →	12	$9006
	BC	$9005
	9A	$9004
	DE	$9003
		$9002
		$9001
		$9000

MEMORY

Examples of Push and Pop Operations

d0	XX	XX	XX	XX
d1	XX	XX	9A	BC
d2	XX	XX	XX	DE

```
move.l d0,-(a1)    ;push long word
move.w d1,-(a1)    ;push word
move.b d2,-(a1)    ;push byte
                .
                .
                .
move.b (a1)+,d2    ;pop byte
move.w (a1)+,d1    ;pop word
move.l (a1)+,d0    ;pop long word
```



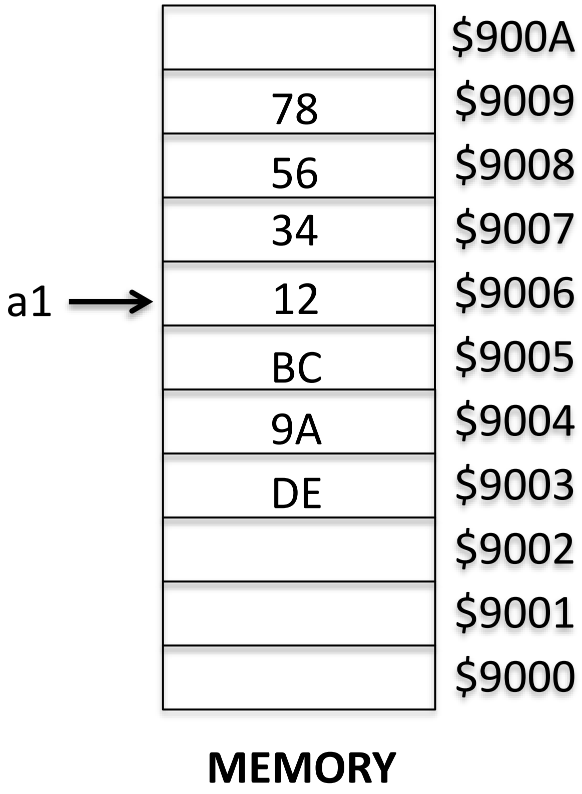
		$900A
	78	$9009
	56	$9008
	34	$9007
a1 →	12	$9006
	BC	$9005
	9A	$9004
	DE	$9003
		$9002
		$9001
		$9000

MEMORY

Examples of Push and Pop Operations

d0	12	34	56	78
d1	XX	XX	9A	BC
d2	XX	XX	XX	DE

```
move.l d0,-(a1)    ;push long word
move.w d1,-(a1)    ;push word
move.b d2,-(a1)    ;push byte
                 .
                 .
                 .
move.b (a1)+,d2    ;pop byte
move.w (a1)+,d1    ;pop word
move.l (a1)+,d0    ;pop long word
```



Examples of Push and Pop Operations

d0	12	34	56	78
d1	XX	XX	9A	BC
d2	XX	XX	XX	DE

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
               .
               .
               .
move.b (a1)+,d2     ;pop byte
move.w (a1)+,d1     ;pop word
move.l (a1)+,d0     ;pop long word
```

a1 →		$900A
	78	$9009
	56	$9008
	34	$9007
	12	$9006
	BC	$9005
	9A	$9004
	DE	$9003
		$9002
		$9001
		$9000

MEMORY

Unbalanced Stack

d0	12	34	56	78
d1	XX	XX	9A	BC
d2	XX	XX	XX	DE

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
                .
                .
                .
move.b (a1)+,d2     ;pop byte
move.l (a1)+,d0     ;pop long word
move.w (a1)+,d1     ;pop word
```

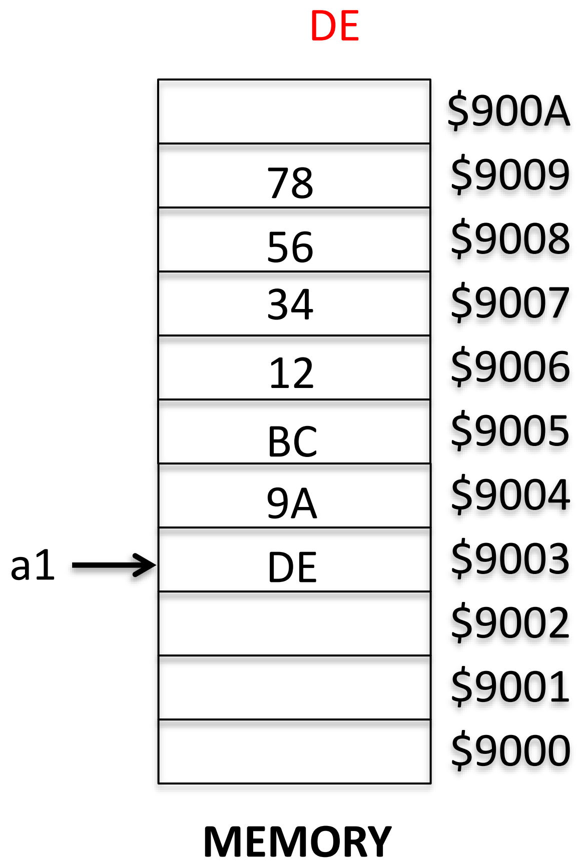
		$900A
	78	$9009
	56	$9008
	34	$9007
	12	$9006
	BC	$9005
	9A	$9004
a1 →	DE	$9003
		$9002
		$9001
		$9000

MEMORY

Unbalanced Stack

d0	12	34	56	78
d1	XX	XX	9A	BC
d2	XX	XX	XX	DE

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
                .
                .
                .
move.b (a1)+,d2     ;pop byte
move.l (a1)+,d0     ;pop long word
move.w (a1)+,d1     ;pop word
```



DE

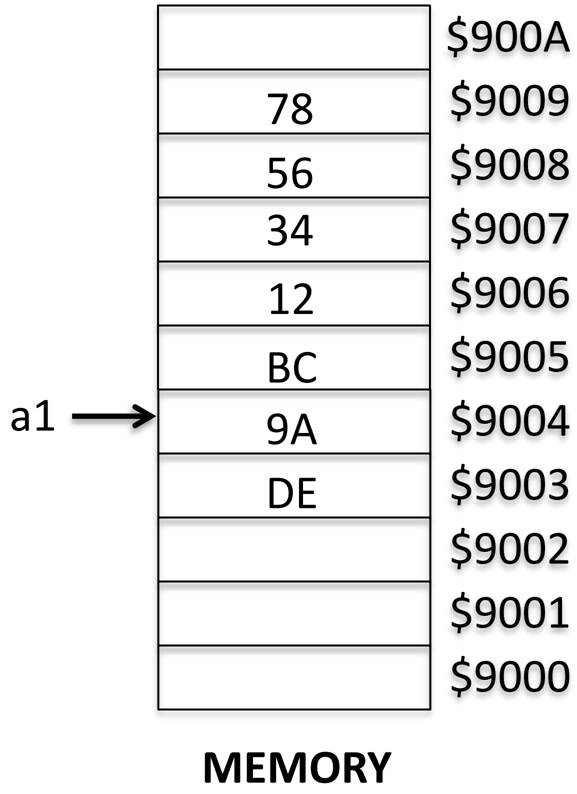
	$900A
78	$9009
56	$9008
34	$9007
12	$9006
BC	$9005
9A	$9004
DE	$9003
	$9002
	$9001
	$9000

MEMORY

Unbalanced Stack

d0	12	34	56	78
d1	XX	XX	9A	BC
d2	XX	XX	XX	DE

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
                .
                .
                .
move.b (a1)+,d2     ;pop byte
move.l (a1)+,d0     ;pop long word
move.w (a1)+,d1     ;pop word
```

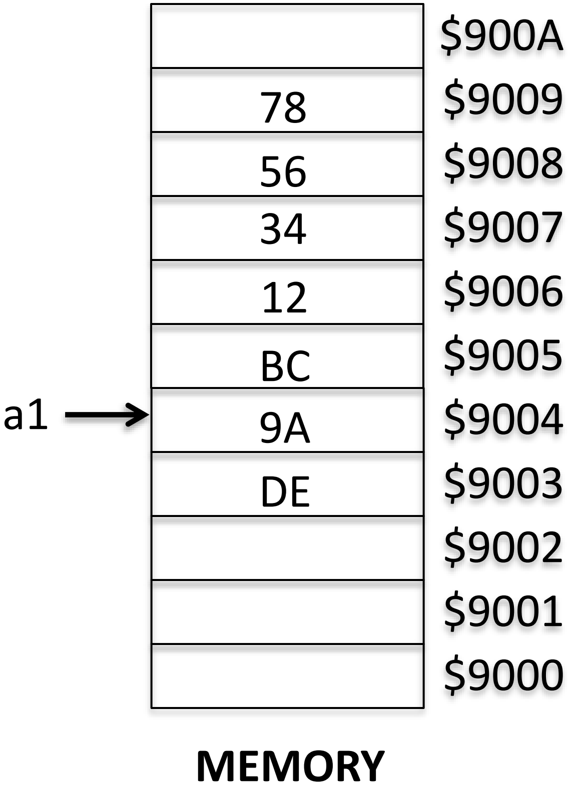


	$900A
78	$9009
56	$9008
34	$9007
12	$9006
BC	$9005
9A	$9004
DE	$9003
	$9002
	$9001
	$9000

Unbalanced Stack

d0	12	34	56	78
d1	XX	XX	9A	BC
d2	XX	XX	XX	DE

```
move.l d0,-(a1)    ;push long word
move.w d1,-(a1)    ;push word
move.b d2,-(a1)    ;push byte
                .
                .
                .
move.b (a1)+,d2    ;pop byte
move.l (a1)+,d0    ;pop long word
move.w (a1)+,d1    ;pop word
```



	$900A
78	$9009
56	$9008
34	$9007
12	$9006
BC	$9005
a1 → 9A	$9004
DE	$9003
	$9002
	$9001
	$9000

MEMORY

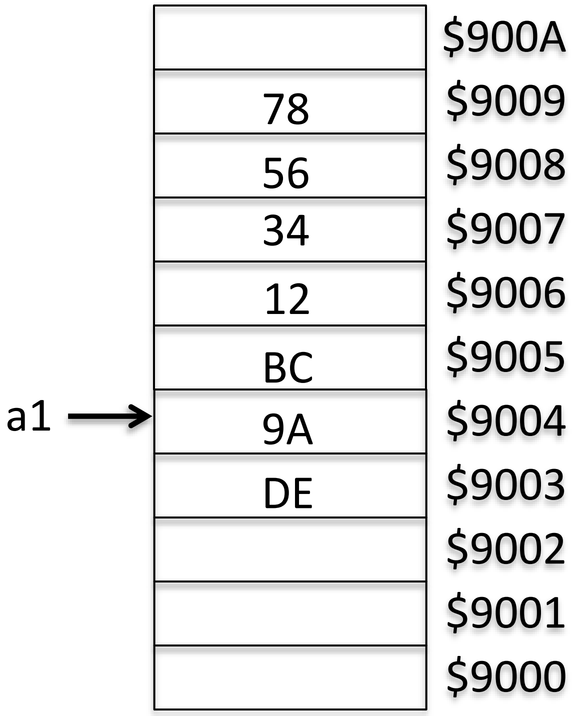
Unbalanced Stack

d0	12	34	56	78
	9A	BC	12	34

d1	XX	XX	9A	BC

d2	XX	XX	XX	DE

```
move.l d0,-(a1)     ;push long word
move.w d1,-(a1)     ;push word
move.b d2,-(a1)     ;push byte
                 .
                 .
                 .
move.b (a1)+,d2     ;pop byte
move.l (a1)+,d0     ;pop long word
move.w (a1)+,d1     ;pop word
```



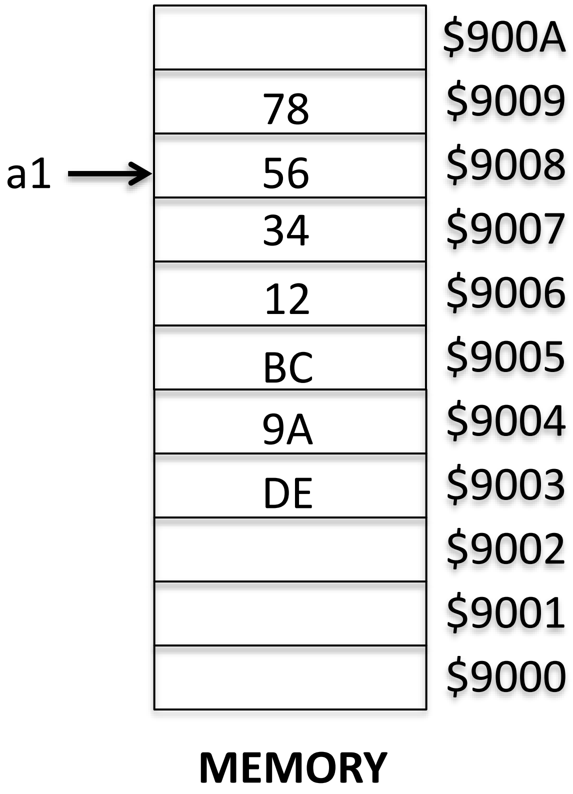
	$900A
78	$9009
56	$9008
34	$9007
12	$9006
BC	$9005
9A	$9004
DE	$9003
	$9002
	$9001
	$9000

MEMORY

Unbalanced Stack

d0	12	34	56	78
	9A	BC	12	34
d1	XX	XX	9A	BC
d2	XX	XX	XX	DE

```
move.l d0,-(a1)    ;push long word
move.w d1,-(a1)    ;push word
move.b d2,-(a1)    ;push byte
                .
                .
                .
move.b (a1)+,d2    ;pop byte
move.l (a1)+,d0    ;pop long word
move.w (a1)+,d1    ;pop word
```



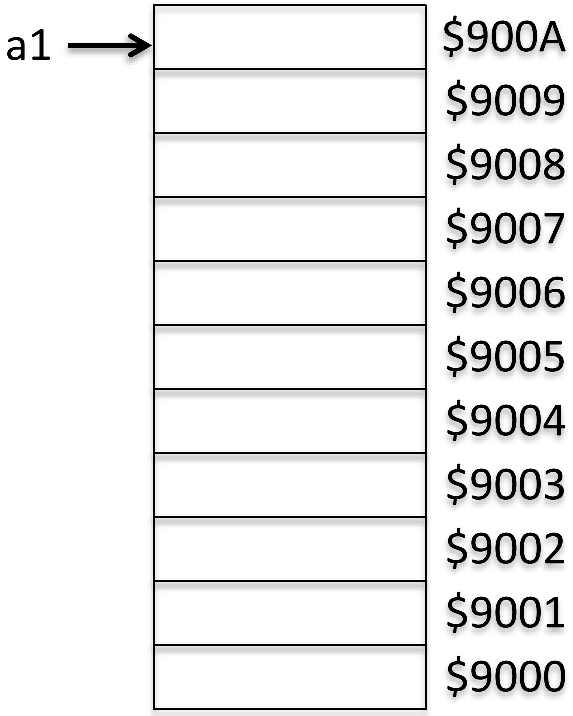
	$900A
78	$9009
a1 → 56	$9008
34	$9007
12	$9006
BC	$9005
9A	$9004
DE	$9003
	$9002
	$9001
	$9000

MEMORY

Address Error Exception

d0	12	34	56	78
d1	XX	XX	9A	BC
d2	XX	XX	XX	DE

```
move.b d0,-(a1)    ;push byte
move.w d2,-(a1)    ;push word
```

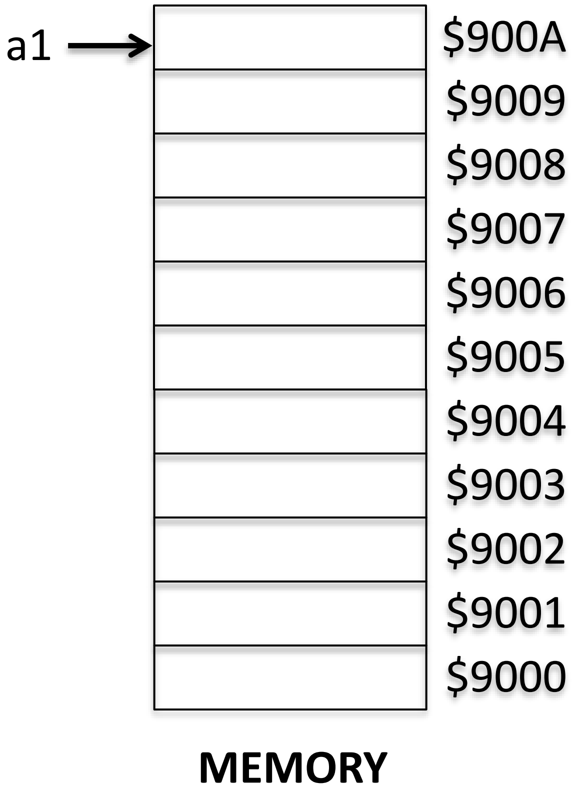


MEMORY

Address Error Exception

d0	12	34	56	78
d1	XX	XX	9A	BC
d2	XX	XX	XX	DE

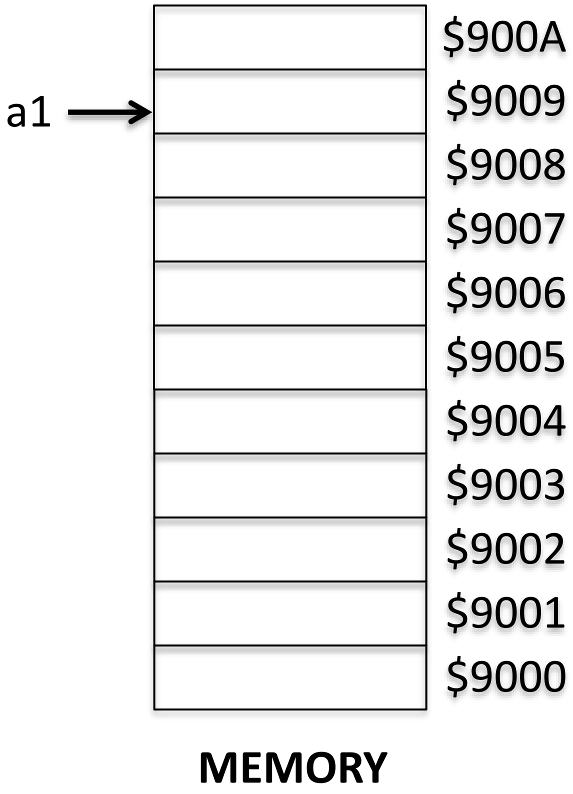
```
move.b d0,-(a1)    ;push byte
move.w d2,-(a1)    ;push word
```



Address Error Exception

d0	12	34	56	78
d1	XX	XX	9A	BC
d2	XX	XX	XX	DE

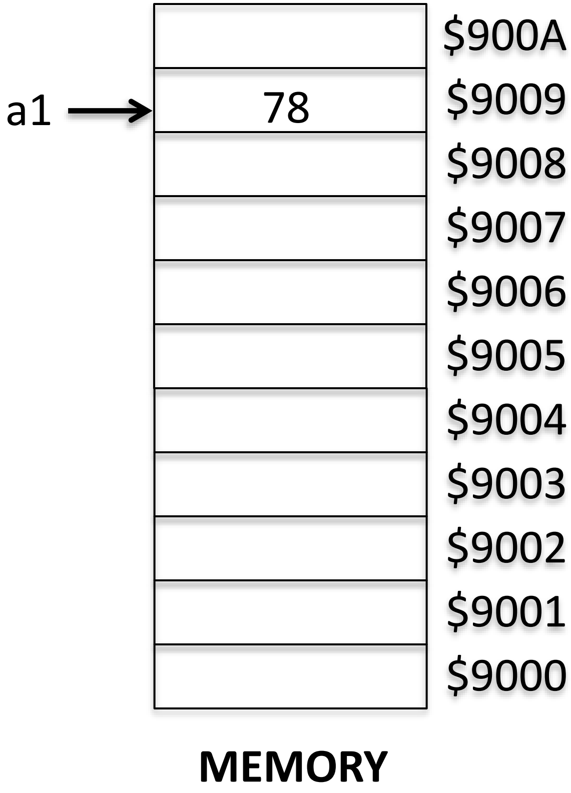
```
move.b d0,-(a1)     ;push byte
move.w d2,-(a1)     ;push word
```



Address Error Exception

d0	12	34	56	78
d1	XX	XX	9A	BC
d2	XX	XX	XX	DE

```
move.b d0,-(a1)     ;push byte
move.w d2,-(a1)     ;push word
```



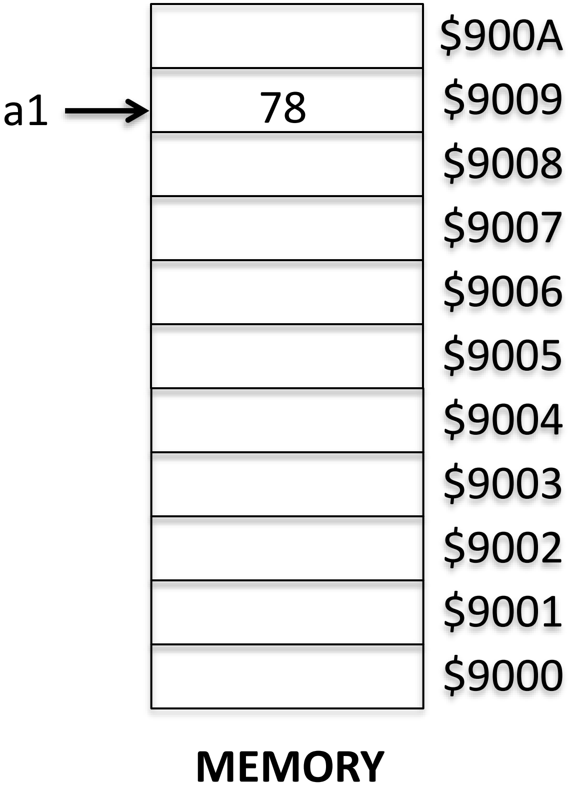
		$900A
a1 →	78	$9009
		$9008
		$9007
		$9006
		$9005
		$9004
		$9003
		$9002
		$9001
		$9000

MEMORY

Address Error Exception

d0	12	34	56	78
d1	XX	XX	9A	BC
d2	XX	XX	XX	DE

```
move.b d0,-(a1)     ;push byte
move.w d2,-(a1)     ;push word
```



a1 → 78

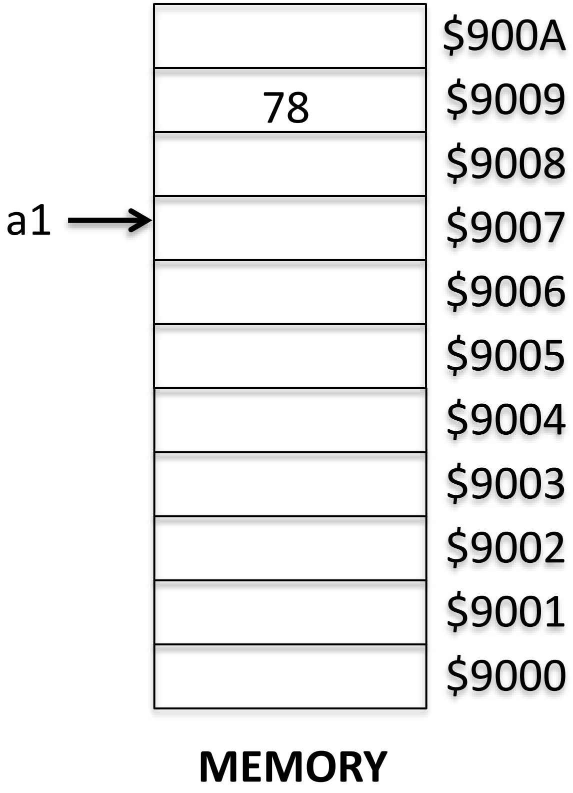
$900A
$9009
$9008
$9007
$9006
$9005
$9004
$9003
$9002
$9001
$9000

MEMORY

Address Error Exception

d0	12	34	56	78
d1	XX	XX	9A	BC
d2	XX	XX	XX	DE

```
move.b d0,-(a1)     ;push byte
move.w d2,-(a1)     ;push word
```

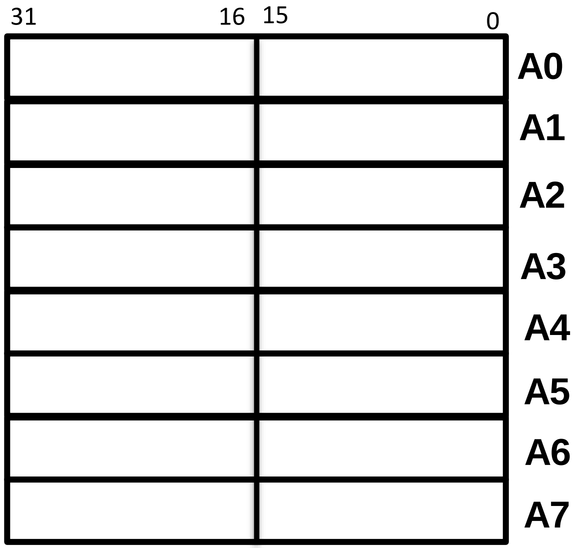


		$900A
	78	$9009
		$9008
a1 →		$9007
		$9006
		$9005
		$9004
		$9003
		$9002
		$9001
		$9000

MEMORY

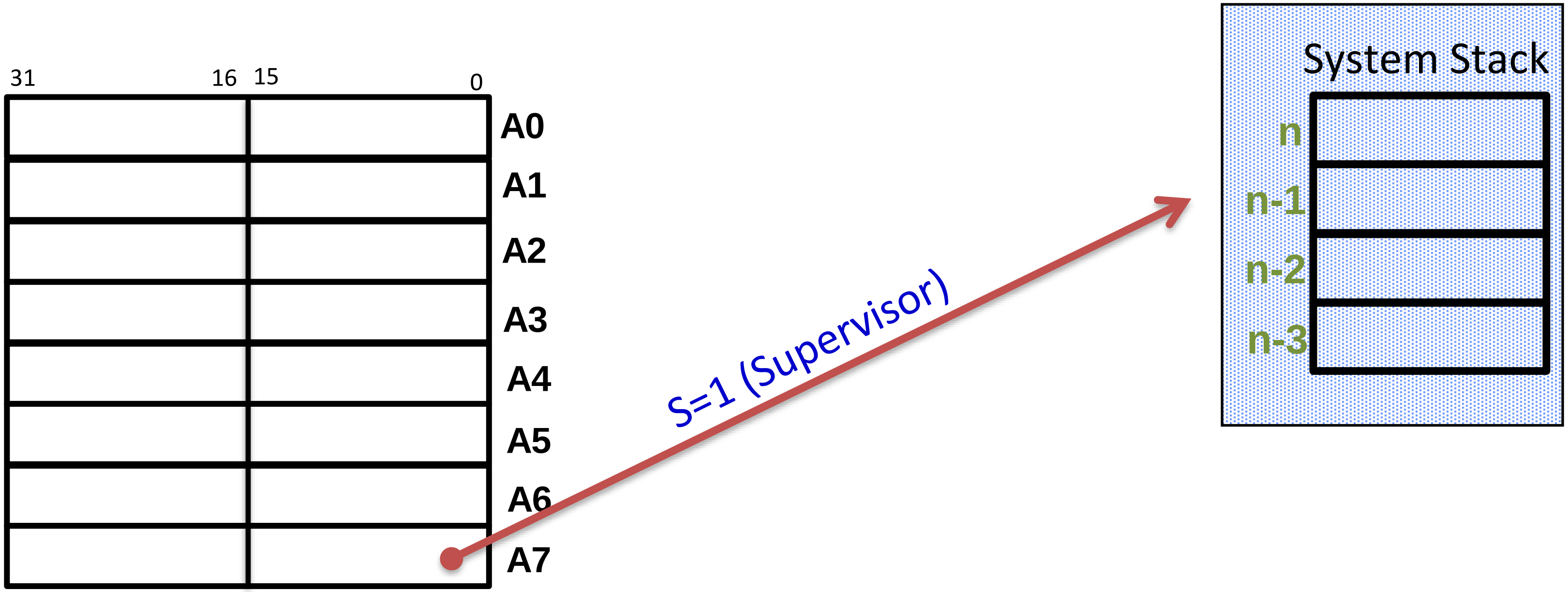
System and User Runtime Stacks and A7

- Address register A7 is used as a dedicated stack pointer



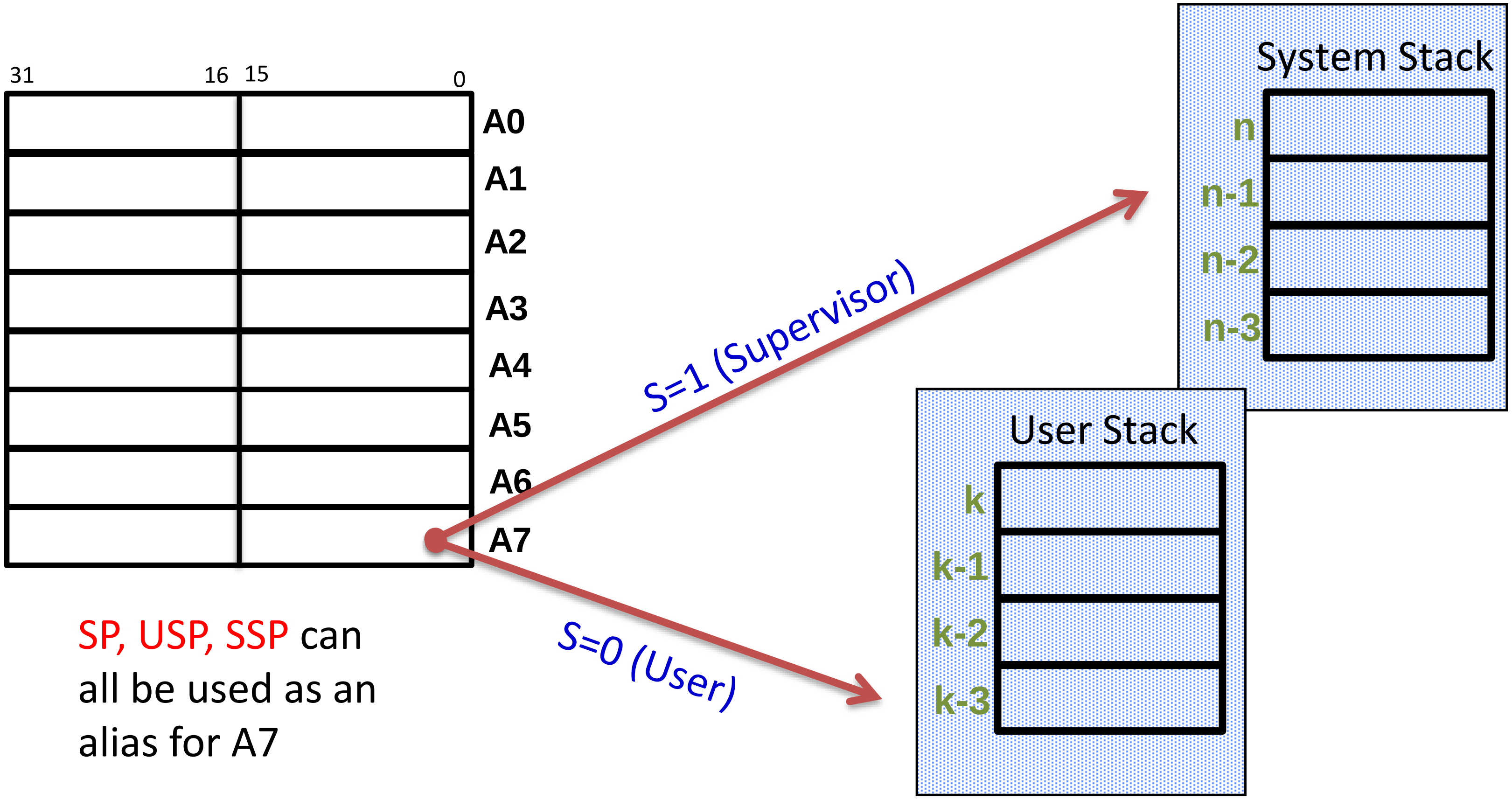
System and User Runtime Stacks and A7

- Address register A7 is used as a dedicated stack pointer



System and User Runtime Stacks and A7

- Address register A7 is used as a dedicated stack pointer



SP, USP, SSP can all be used as an alias for A7

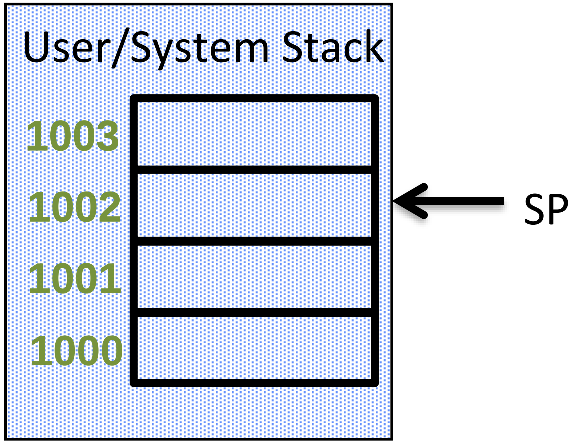
Common Stack Operations

Operation	Syntax	Description
Push to hardware stack	`move <ea>,-(sp)`	Decrement SP, then move <ea> to stack
Pop from hardware stack	`move (sp)+,<ea>`	Move information from stack to <ea>, then increment SP
Push Address to hardware stack	`pea <ea>`	<ea> is computed and pushed to stack
Move to stack	`move <ea>,d(sp)`	Move <ea> to SP + d without changing SP
Move from stack	`move d(sp),<ea>`	Move from SP + d to <ea> without changing SP

Runtime Stack and Byte Operations

- Word boundaries must be maintained on the hardware stack
  - if the stack pointer is A0-A6
    - the pointer is incremented or decremented by 1 when performing a byte operation
  - if the stack pointer is A7 (SP)
    - the pointer is incremented or decremented by 2 when performing a byte operation

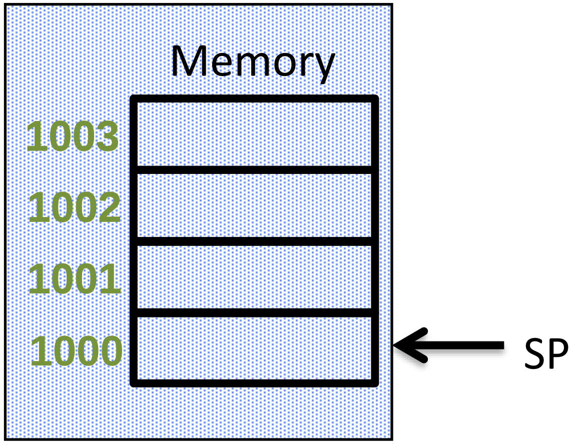
```
move.b #$ff,-(sp)
```



Runtime Stack and Byte Operations

- Word boundaries must be maintained on the hardware stack
  - if the stack pointer is A0-A6
    - the pointer is incremented or decremented by 1 when performing a byte operation
  - if the stack pointer is A7 (SP)
    - the pointer is incremented or decremented by 2 when performing a byte operation

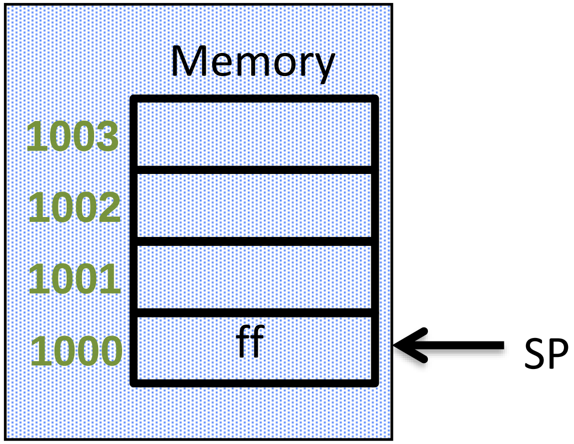
```
move.b #$ff,-(sp)
```



Runtime Stack and Byte Operations

- Word boundaries must be maintained on the hardware stack
  - if the stack pointer is A0-A6
    - the pointer is incremented or decremented by 1 when performing a byte operation
  - if the stack pointer is A7 (SP)
    - the pointer is incremented or decremented by 2 when performing a byte operation

```
move.b #$ff,-(sp)
```

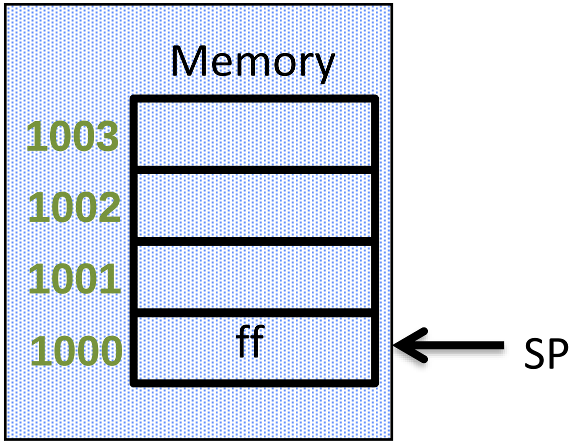


Runtime Stack and Byte Operations

- Word boundaries must be maintained on the hardware stack
  - if the stack pointer is A0-A6
    - the pointer is incremented or decremented by 1 when performing a byte operation
  - if the stack pointer is A7 (SP)
    - the pointer is incremented or decremented by 2 when performing a byte operation

```
move.b (sp)+,d0
```

d0	XX	XX	XX	XX

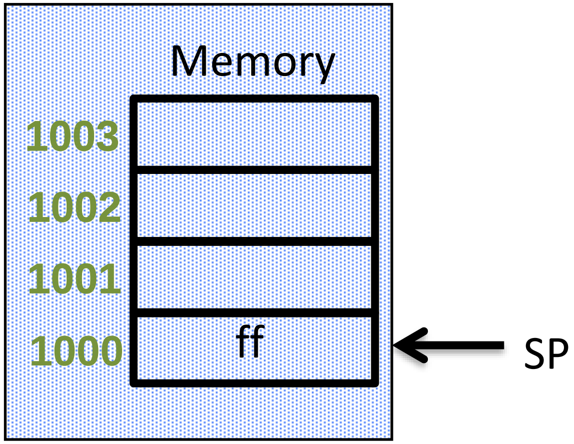


Runtime Stack and Byte Operations

- Word boundaries must be maintained on the hardware stack
  - if the stack pointer is A0-A6
    - the pointer is incremented or decremented by 1 when performing a byte operation
  - if the stack pointer is A7 (SP)
    - the pointer is incremented or decremented by 2 when performing a byte operation

```
move.b (sp)+,d0
```

d0	XX	XX	XX	ff



Memory

1003

1002

1001

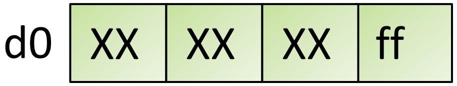
1000 ff

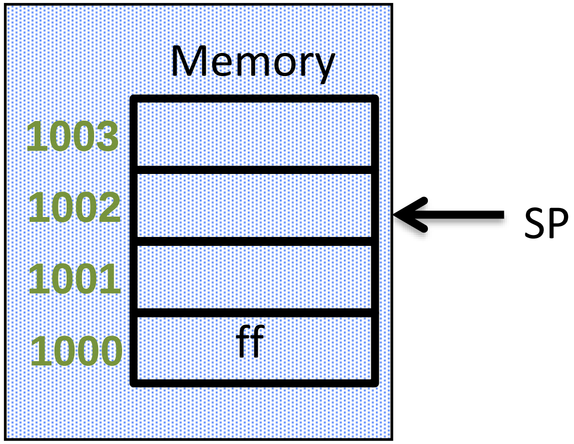
SP

Runtime Stack and Byte Operations

- Word boundaries must be maintained on the hardware stack
  - if the stack pointer is A0-A6
    - the pointer is incremented or decremented by 1 when performing a byte operation
  - if the stack pointer is A7 (SP)
    - the pointer is incremented or decremented by 2 when performing a byte operation

```
move.b (sp)+,d0
```





Summary

- Stacks grow from the “top” of memory towards the “bottom” of memory
- 68000 has two A7 registers – each acts as a stack pointer for one of two runtime stacks
  - User (S=0) and System (S=1)
    - Two stacks protects OS from user(s)
  - Word boundaries must be maintained
  - Stack pointer is incremented by 2 on byte operations
- Push and Pop operations
  - Synthesized by combining MOVE with pre-decement and post-increment addressing
  - Other operations allowed on stack (useful when implementing functions in high-level languages)