香港中文大學
The Chinese University of Hong Kong

CSCI2510 Computer Organization

Tutorial 08: Direct Mapping in MASM

Yuhong LIANG

yhliang@cse.cuhk.edu.hk

Outline

- Assignment 2 Solution
- Direct mapping implementation

Assignment 2 Solution

Question:

List the common flags in Condition Code Register, and describe the situation when they are set to 1 and give an example for each flag.

Answer:

Given two 4-bit registers R1 and R2 storing signed integers in 2's-complement format. Take add R1, R2 as example.

(1)N (negative) Set to 1 if the result is negative

$R1 = 2$, $R2 = -5$

(2)Z (zero) Set to 1 if the result is 0

$R1 = 2$, $R2 = -2$

(3)V (overflow) Set to 1 if arithmetic overflow occurs

$R1 = 7$, $R2 = 1$

(4)C (carry) Set to 1 if a carry-out occurs

$R1 = 5$, $R2 = -2$

Assignment 2 Solution

Type of Operands: Address Modes (1/2)

- **Addressing Modes**: the ways for specifying the locations of instruction operands.

Address Mode	Assembler Syntax	Addressing Function
1) Immediate	$\#Value$	$Operand = Value$
2) Register	Ri	$EA = Ri$
3) Absolute	LOC	$EA = LOC$
4) Register indirect	(Ri)	$EA = [Ri]$
5) Index	$X(Ri)$	$EA = [Ri] + X$
6) Base with index	(Ri, Rj)	$EA = [Ri] + [Rj]$

EA: effective address
Value: a signed number
X: index value

Assignment 2 Solution

Question:

Determine the effective address (EA) of the last operand, given R1 = 1024, R2 = 512, R3 = 256, and the memory location LOC = 1024

(a) ADD R1, R2, R3

(b) LOAD R1, (R3)

(c) LOAD R1, (R2, R3)

(d) LOAD R1, LOC

(e) LOAD R1, -128(R1)

Answer:

(a) R3

(b) 256

(c) 768

(d) 1024/LOC

(e) 896

Assignment 2 Solution

- If inputnumber is equel to 0,do the pop action
- If inputnumber is not equel to 0, push that inputnumebr

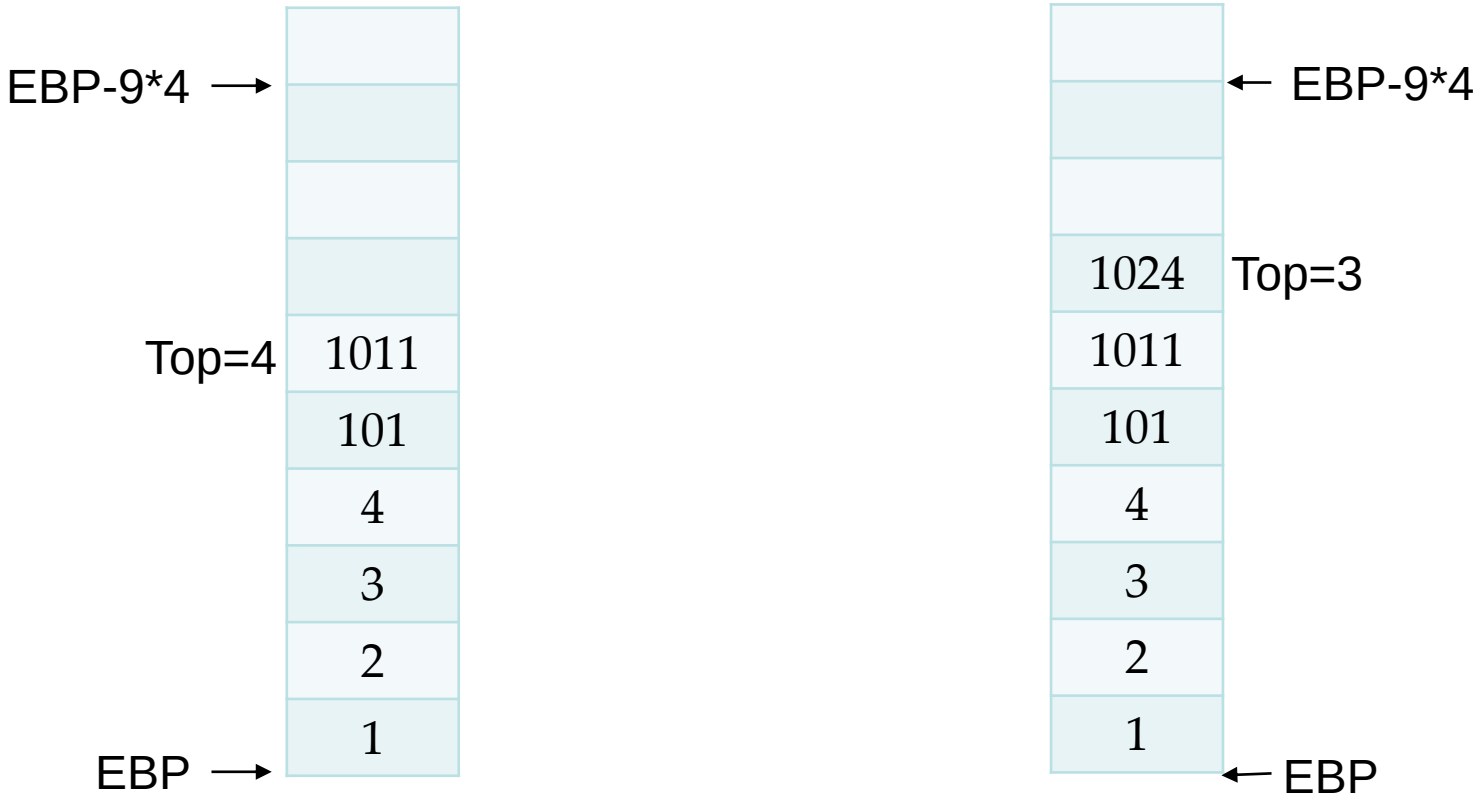
```
input:
    invoke crt_printf, addr inputStatement
    invoke crt_scanf, addr numberFormat, addr inputnumber ; input number
    mov ECX, inputnumber ; load the inputnumber in register ECX
    cmp ECX, 0 ; compare content of ECX with 0(fill it)
    je popnumber ; if content of ECX is equal to 0, then jump to popnumber(inputnumber 0 repres
    jmp pushnumber ; jmp to pushnumber(fill it)
```

- Print errror

```
poperror:
    invoke crt_printf, addr popErrorStatement ; print error message(fill it)
    jmp exitprogram
```

Assignment 2 Solution

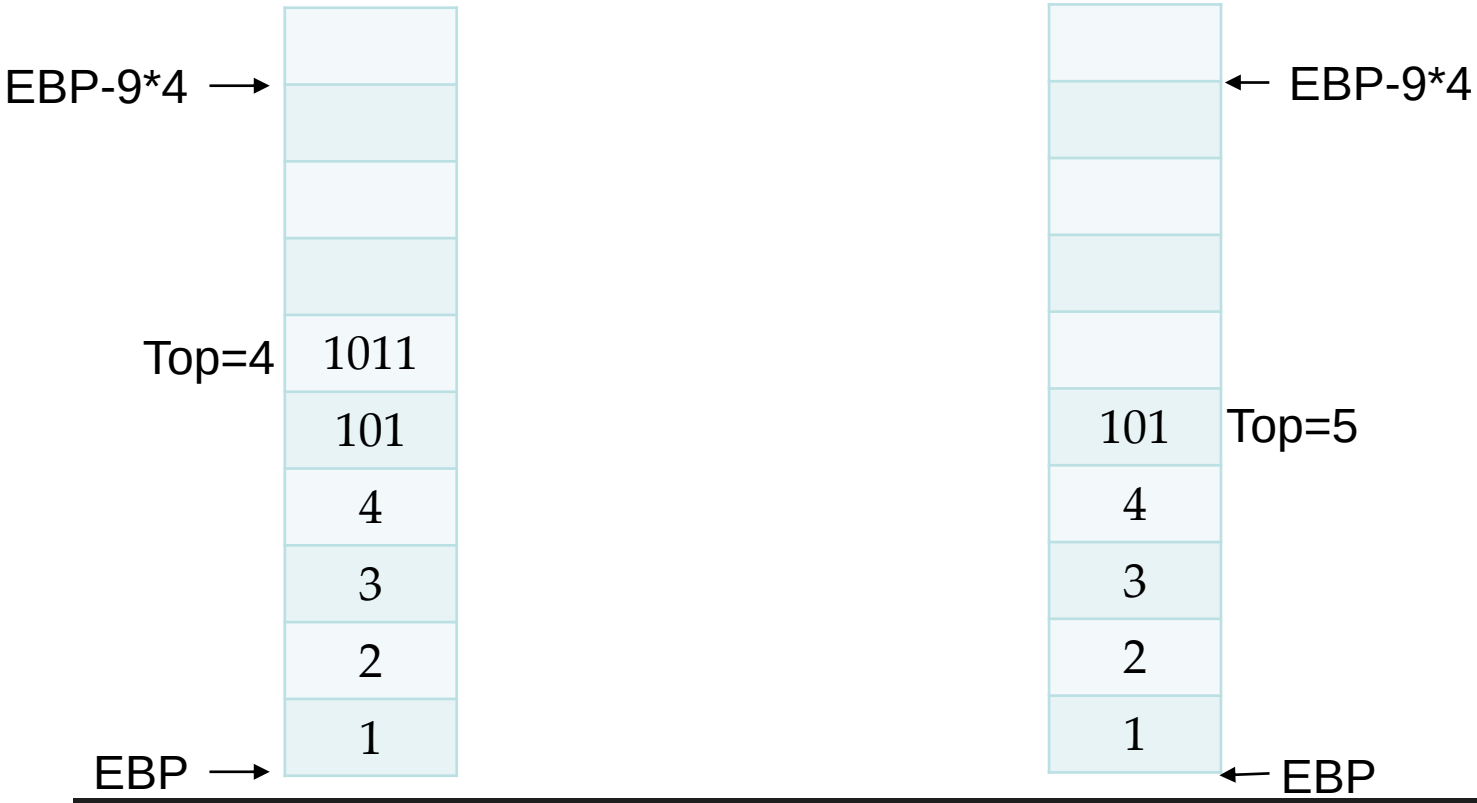
push




```
pushnumber:
    mov EAX, top ; load the top pointer to EAX
    cmp EAX, 0 ; see if stack is full
    je pusherror ; if stack is full, jump to pusherror
    sub EAX, 1 ; sub the pointer(fill it)
    mov top, EAX ; store the pointer to memory
    mov ECX, inputnumber
    mov [EBP - 4 * 10 + 4 * EAX], ECX ; push the inputnumber in stack in memory(fill it)
    jmp input
```

Assignment 2 Solution

pop

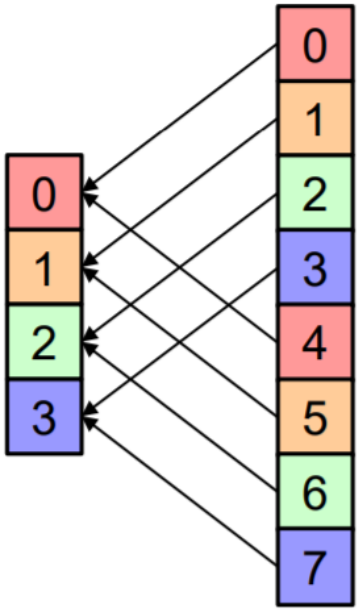



```
popnumber:
    mov EAX, top ; load the top pointer to EAX
    mov EBX, stacklength ; load the stack length to EBX
    cmp EAX, EBX ; see if the stack is empty(fill it)
    je poperror ; if the stack is empty jump to poperror(fill it)
    mov ECX, [EBP - 4 * 10 + 4 * EAX] ; get the top data of the stack, and load it to ECX
    invoke crt_printf, addr outputFormat, ECX ; print out the top data
    mov EAX, top ; load the top pointer to EAX
    add EAX, 1 ; add the pointer(fill it)
    mov top, EAX ; store the pointer to memory(fill it)
    jmp input
```

Direct mapping implementation

Direct

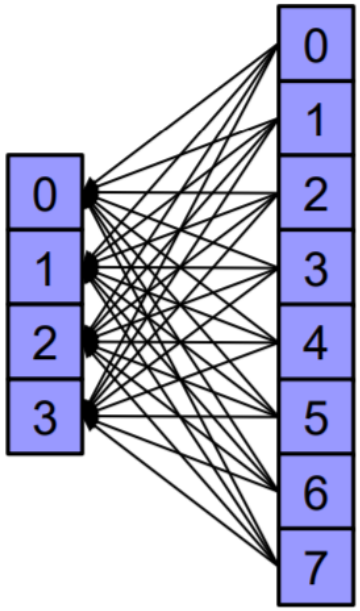
A Memory Block is **directly mapped** (%) to a Cache Block.



Associative

A Memory Block can be **mapped to any** Cache Block.

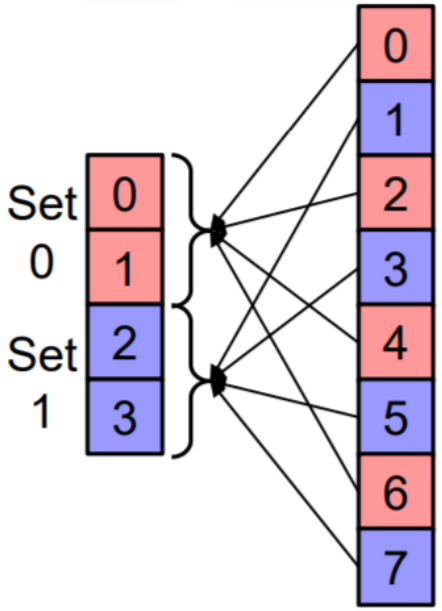
(First come first serve!)



Set Associative

A Memory Block is **directly mapped** (%) to a **Cache Set**.

In a **Set**? **Associative**



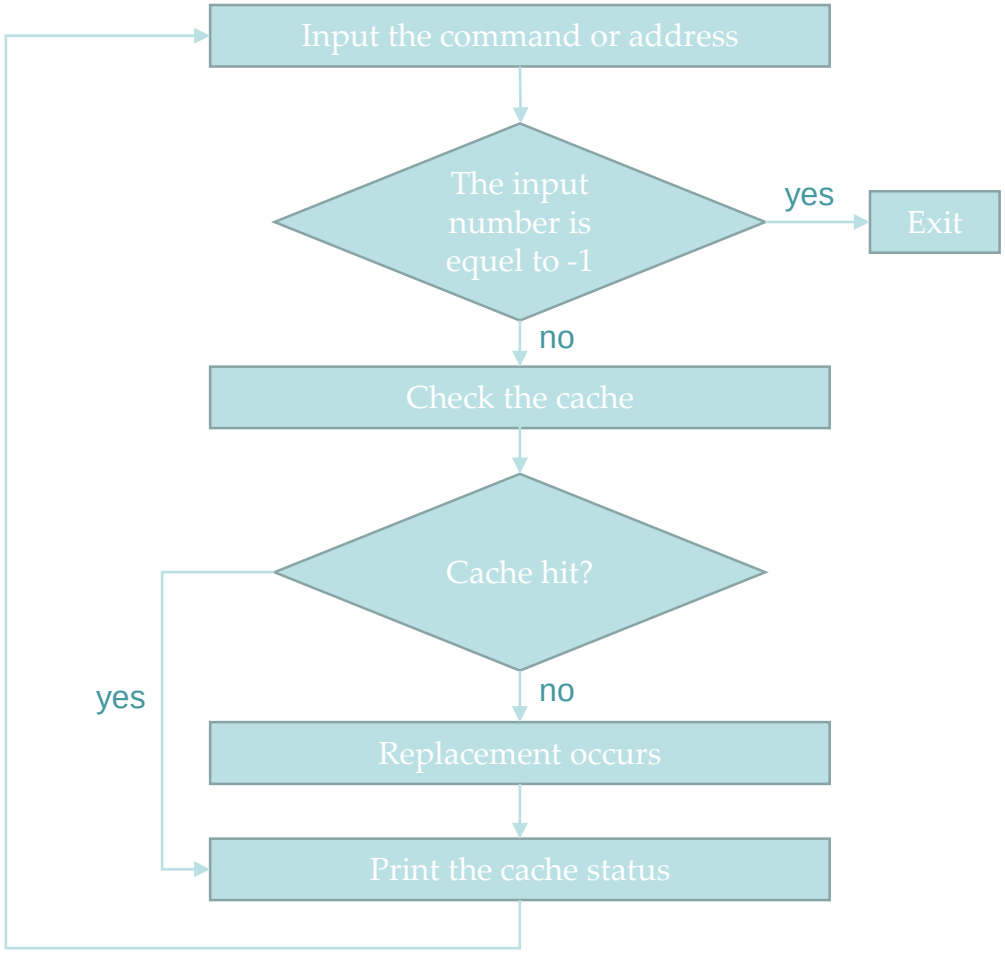
Direct mapping implementation

procedure

- Check the cache
- Replace the block

CPU Access →	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
Status of cache		**0**	**0**	**0**		**0**		**4**			**0**							**0**		
			1	1		1		1			1							1		
				2		2		2			2							2		
	7	7	7	7		3		3			3							7		

Direct mapping implementation



Direct mapping implementation

- Data definition:
- Cache size=4
- CPUAccess type:dd(32 bit)

```
CacheBlocks dd 4 dup(-1)

cacheSize dd 4, 0 ;
CPUAccess dd "%d", 0
```

- Check the cache and replace

```
input:
    invoke crt_printf, addr inputStatement
    invoke crt_scanf, addr CPUAccessFormat, addr CPUAccess
    mov ECX, CPUAccess
    cmp ECX, -1
    je exitprogram
    jmp checkcacheorreplace

checkcacheorreplace:
    mov EAX, CPUAccess
    mov EDX, 0
    idiv cacheSize
    mov EAX, CPUAccess
    cmp EAX, [EBP + EDX*4]
    je printcachestatus
    mov [EBP + EDX*4], EAX
    jmp printcachestatus
```

Summary

- Assignment 2 Solution
- Direct mapping implementation