

Assignment NO. 2.

D	D	M	M	Y	Y	Y

(Q1) Explain different ways of specifying target address in control transfer instructions with diagram.

The target address in a control transfer instruction can be specified in one or the following ways.

(a) Immediate constant: In this method, target address is specified along with instruction and immediate data.

Eg.

jmp 0x100

Target address = A + 0x100

where A = Address of next instruction

(b) jmp *0x100

target address = 0x100

(b) Register Addressing:

In this method target address is specified using a register.

Eg.

jmp %eax

target address = Address of eax

jmp %ebx

(c) Memory Addressing Mode:

In this method target instruction is read from memory.

Eg.

jmp (%eax)

Here address of target location instruction is stored in memory location whose address is in regular register eax.

D	D	M	M	Y	Y	Y	Y
---	---	---	---	---	---	---	---

Q2. Write short note on function call and return.

→ Functions are sub programs to which the control of execution is passed temporarily. After executing of the called function is completed, the control is transferred back to the calling program.

The most important reason is to reuse the code. Example: If in a program several strings are to be displayed on the screen at different time, it is better to write a function to display, this function can be called.

Writing function in a program to make it readable and understandable.

Example: CALL FUNCTION

0x100C PUSH Y.EAX

0x1100 FUNCTION:

During execution of CALL instruction, address of next instruction is saved on stack. Stack pointer is decremented by 4. The register EIP changed to 0x000100 there by control transfer to function FUNCTION. IA 32 Architecture provide an instruction called RET to return control to the caller after after execution of the called function is completed.

The RET instruction pops a 32 bit offset from top of the stack into EIP register thereby causing a control transfer.



Q3. Write 80386 assembly language program for function call and return.

.Section .data

output

.ascii "This is section layd in !"

.section .text

.global _start

_start: pushl \$1

pushl \$output

call printf

addl \$8, %esp

call function

pushl \$3d

pushl \$output

call printf

addl \$8, %esp

more \$1, %eax

more \$0, %ebx

int \$0x80

Function: pushl %ebp

morel %esp, %ebp

pushl \$20

pushl \$output

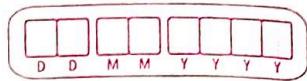
call printf

addl \$8, %esp

morel %ebp, %esp

popl %ebp

ret.



Q4. Write 80386 assembly language program to find largest number from an array of 32 bit numbers.



.section .data

value:

.int 100, 56, 130, 65, 123

.section .text

.global _start

_start: .nop

more value, %eax

more \$1, %edi

back: more value(%edi,%eax), %ebx

cmpl %eax, %ebx

cmple %ebx,%eax

ine %edi

cmpl \$5, %edi

jne back

more \$1, %eax

more \$0, %ebx

int \$0x80

Q5. The values of registers are given to be all 0 prior to the execution of the following sequence of instructions.

push 40x487

push 10x833

push 10ffecccdd

inc %esp

pop %ax

{ inc %esp

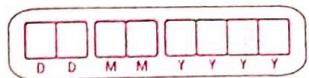
pop %cx

} find value of

ax,cx and esp

after execution

of all instruction



→ Assuming, esp (stack pointer initially 100)

dd	92	92 50 60 Before execution starts.
cc	93	esp = 100 (containing Garbage value)
ee	94	
ff	95	ax=0
33	96	cx=0
00	97	
87	98	
04	99	
xx	100	

(i) pushl \$0x487

pushes 32 bit data on the stack and stack pointer is decremented by 4.

$$\therefore \text{esp} = 98$$

(ii) pushl \$0x33

pushes 32 bit data on the stack and stack pointer is decremented by 4.

$$\therefore \text{esp} = 96$$

(iii) pushl \$0xfcccccd

pushes 32 bit data on the stack and stack pointer is decremented by 4.

$$\therefore \text{esp} = 92$$

(iv) incq esp

Increments stack pointer by 1.

$$\therefore \text{esp} = 93$$

D	D	M	M	Y	Y	Y
---	---	---	---	---	---	---

(vi) `pop %ax`

pops out 16 bit data from top of stack and stores to ax register and esp incremented by 2
 value at $ax = \text{eecc}$ and $esp = 98$

(vii) `inc %esp` \rightarrow $esp = 96$

(viii) `pop %cx` \rightarrow $cx = \text{033}$ and $esp = 98$

After Execution :

$ax = \text{0xeecc}$, $cx = \text{0x33}$, $esp = 98$

Q7. Explain following instruction with instruction example

(a) `cmovebe`

`cmovebe src, dest`

If ($dest \leq src$) move the content of source to destination

source can be register or memory but destination is always register. This instruction is used immediately after compare instruction

Eg: `cmovebe %eax, %ebx.`

(b) `xchg`

`xchg V1, V2`

It exchanges the contents of $V1$ and $V2$.

$V1$ and $V2$ can be registers or memory locations.

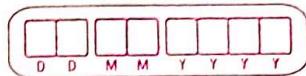
Eg: (i) `xchg %eax, %ebx` (Register to Register transfer)

(ii) `xchg %eax, (%esi)`

or

`xchg (%esi), %eax`

(Register \leftrightarrow memory transfer)



(c) bswap

bswap %eax

It converts big endian type data to little endian type, because Intel doesn't support big endian.

Eg: value = 0x11223344

Before bswap : 44 | 33 | 22 | 11

After bswap : 11 | 22 | 33 | 44
 3 2 1 0

(d) cmpl

cmpl src dest.

If dest - src > 0, carry flag is reset else carry flag is set

i.e. if dest - src > 0 \rightarrow cf = 0

dest - src < 0 \rightarrow cf = 1.

Register values are not changed after execution, only flags are updated.

Eg: cmpl %eax,%ebx

(e) cbtw:

This instruction converts byte (8bit) to word (16bit). sign bit (7th bit) of the source is extended

Eg: Before execution, value = 1111 0000

After execution, value = 1111 1111 1111 0000

(f) pushl: pushl src dest

The source operand of push instruction can be an immediate data, registers, or memory variable.

The source contents are located on the stack & stack pointer (esp) is decremented.



9) popl : popl dest.

The operand can be only registers or memory variable.

The value at the top of the stack is written to destination and stack pointer is incremented.

Eg: popl %eax, %eax

Q8. Write assembly language program to demonstrate function parameter passing by address and reference.



.section .data

num1:

.int oxff

num2:

.int oxee

.section .text

.globl _start

_start:

nop

pushl \$num1,%eax

pushl \$num2,%ebx

call exchange

addl \$2,%eax

movl %eax,%eax

movl \$0,%ebx

int \$0

exchange:

pushl %ebp

movl %esp,%ebp

movl %ebp,%esp

movl %esp,%ebp



xchq ('.eax'), ' .ecx' ;
xchq ('.ebx'), ' .ecx' ;
xchq ('.eax'), ' .ecx' ;
movl ' .ebp', ' .esp' ;
popl ' .ebp' ;
ret

2018-03-20 10:30:00 - 2018-03-20 10:30:00

motif which oral language extends with in

1. 1990 年 10 月 1 日，中国正式成为世界贸易组织成员。

12 ft. 24000 sq ft - total area 16000 sq ft.

1990-1991
1991-1992
1992-1993

1. Ed AFSF information of b26U : AFSA 271112.0
2. Ed FDS & Engineering info of b26U : AFSA 256102.0

• **What is the relationship between the number of hours worked and the total weekly income?**