COMPUTING MACHINERY I

CPSC 355

Raheela Afzal

ICT 524

raheela.afzal1@ucalgary.ca

http://pages.cpsc.ucalgary.ca/~raheela.afzal1/cpsc355/

Memory

Store Register in Memory	
str	stores a register in memory
strh	stores the lower 16-bits of a register in memory, a ***h****alfword*
strb	stores the lower 8-bits of a register in memory, a ***b****yte*

Memory

Read a register from memory	
ldr	loads a register from memory
ldrh	loads the lower 16-bits of a register from memory
ldrb	loads the lower 8-bits of a register from memory
ldrsw	loads a word into a register with sign extend
ldrsh	loads the lower 16-bits of register from memory, with sign extend
ldrsb	loads the lower 8-bits of a register from memory, with sign extend

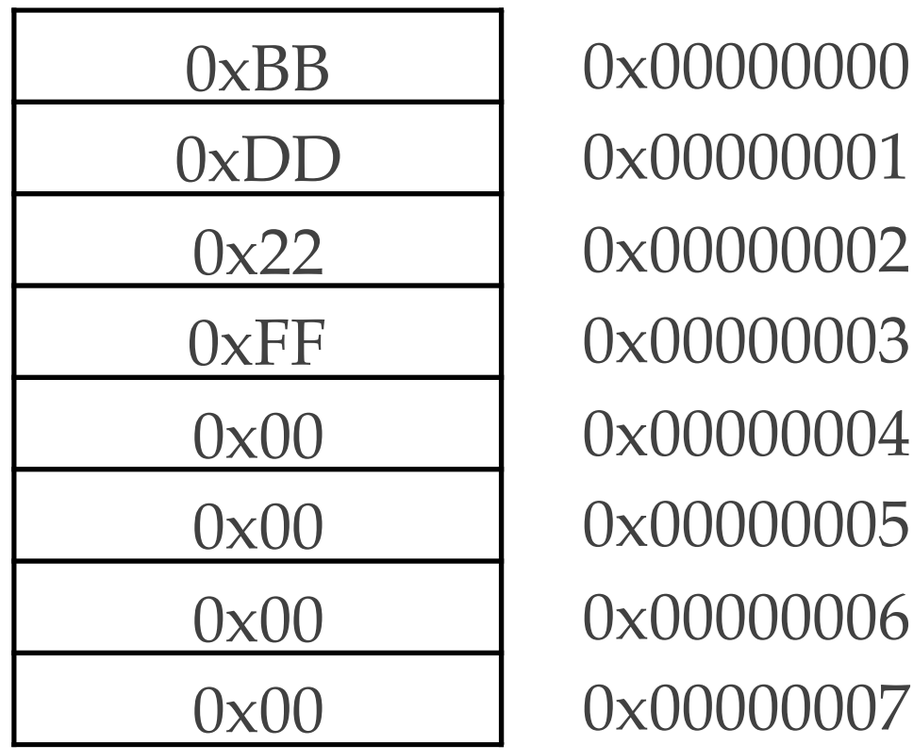
Memory

STORE REGISTER TO MEMORY
64-BIT REGISTER

```
mov x1, wzr
ldr x2, =0xFF22DDBB
str x2, [x1]
```

LOAD REGISTER FROM MEMORY
64-BIT REGISTER

```
mov x1, wzr
ldr x2, [x1]
```



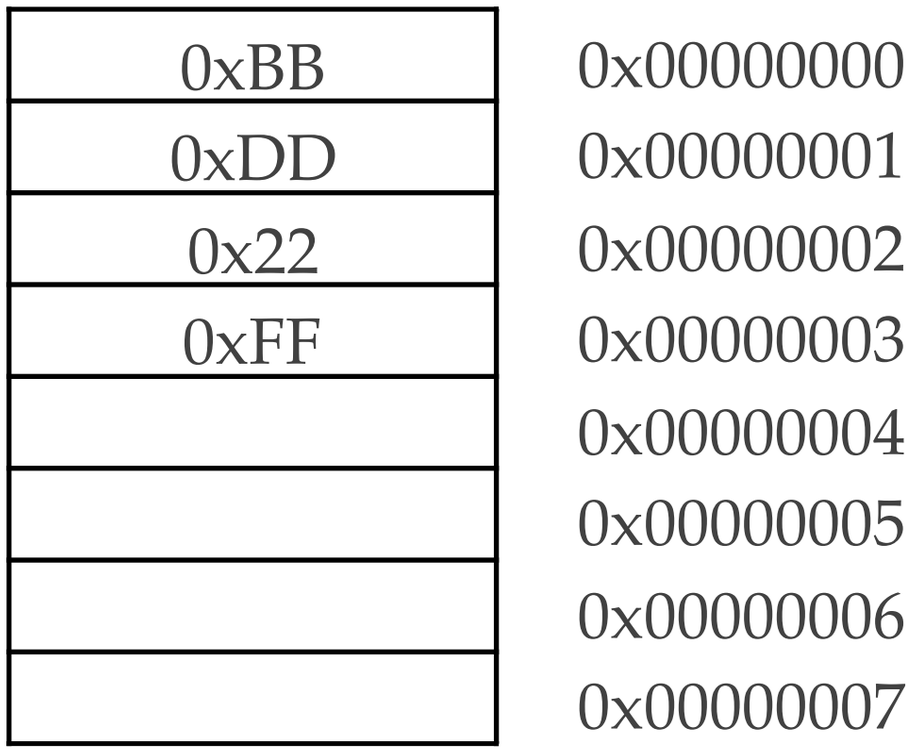
Memory

STORE REGISTER TO MEMORY
32-BIT REGISTER

```
mov x1, wzr
ldr w2, =0xFF22DDBB
str w2, [x1]
```

LOAD REGISTER FROM MEMORY
32-BIT REGISTER

```
mov x1, wzr
ldr w2, [x1]
```



Addressing Modes

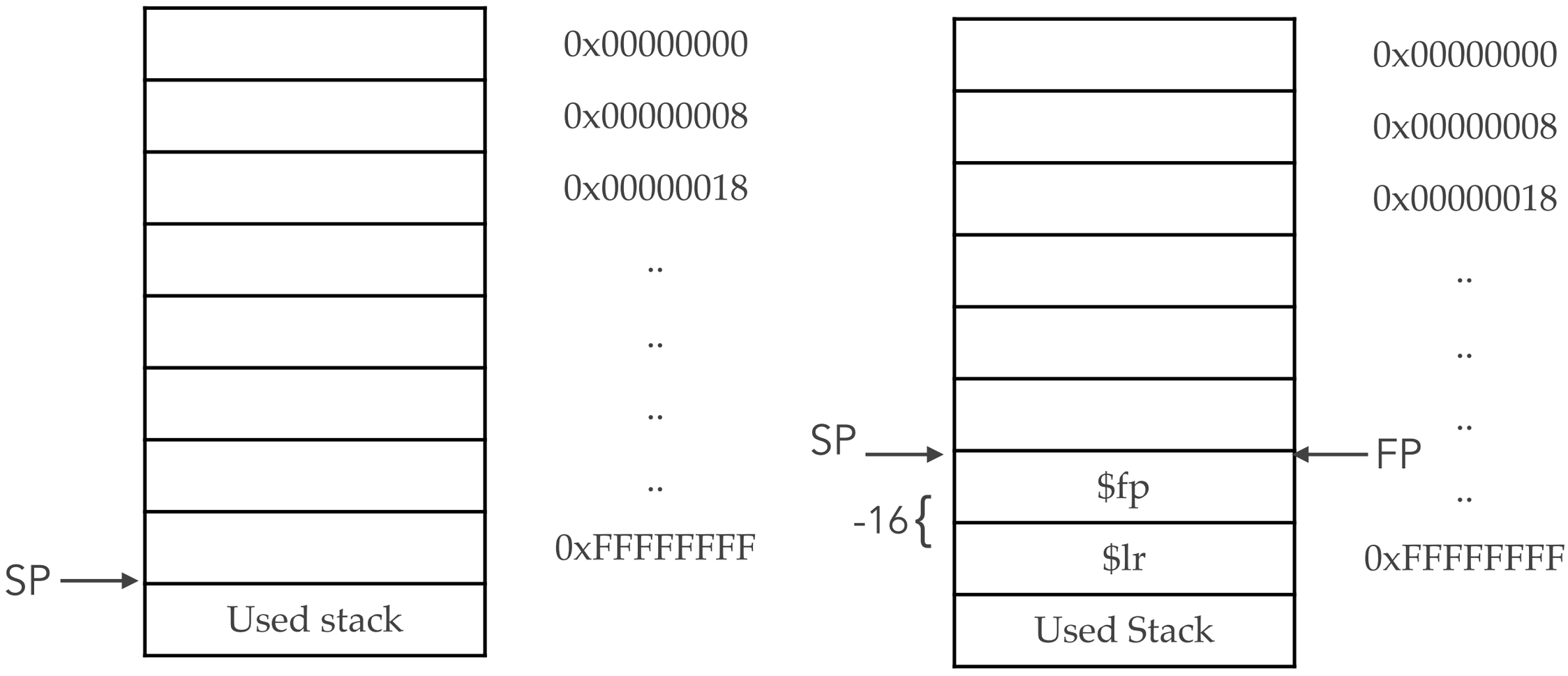
Immediate Offset	str x19, [fp, x22] str x19, [fp, -8]	Address Accessed = fp+offset *fp remains unchanged*
Pre-Increment	str x19, [fp, -8]!	Address Accessed = fp-8 *fp = fp-8*
Post-Increment	str x19, [fp], -8	Address Accessed = fp *fp = fp-8*
Using 32-bit register as offset	str x19, [fp, w22, sxtw]	Address Accessed = fp+(long int)w22 *fp remains unchanged*

Load/Store addressing modes in the A64 instruction set require a 64-bit base address from a general-purpose register or the current stack pointer, SP, with an optional immediate or register offset.

Note: You can also use macros or equates as offsets

The Stack

stp fp, lr, [sp, -16]!
mov fp, sp



PRESERVING 16-BYTE ALIGNMENT OF THE STACK

```
size = 4+4
total_size = 16 + size
alloc = -(total_size) & -16
```

total_size = 24 bytes

-(total_size) = -(11000)

= 11101000 (in 2's complement)

-(total_size) & -16 = 1…11101000

& 1…11110000

1…11100000 (-32)

Alloc = -32 (decrement stack pointer by 32 bytes required for the program)

Dealloc = 32 (move stack pointer back to original address)

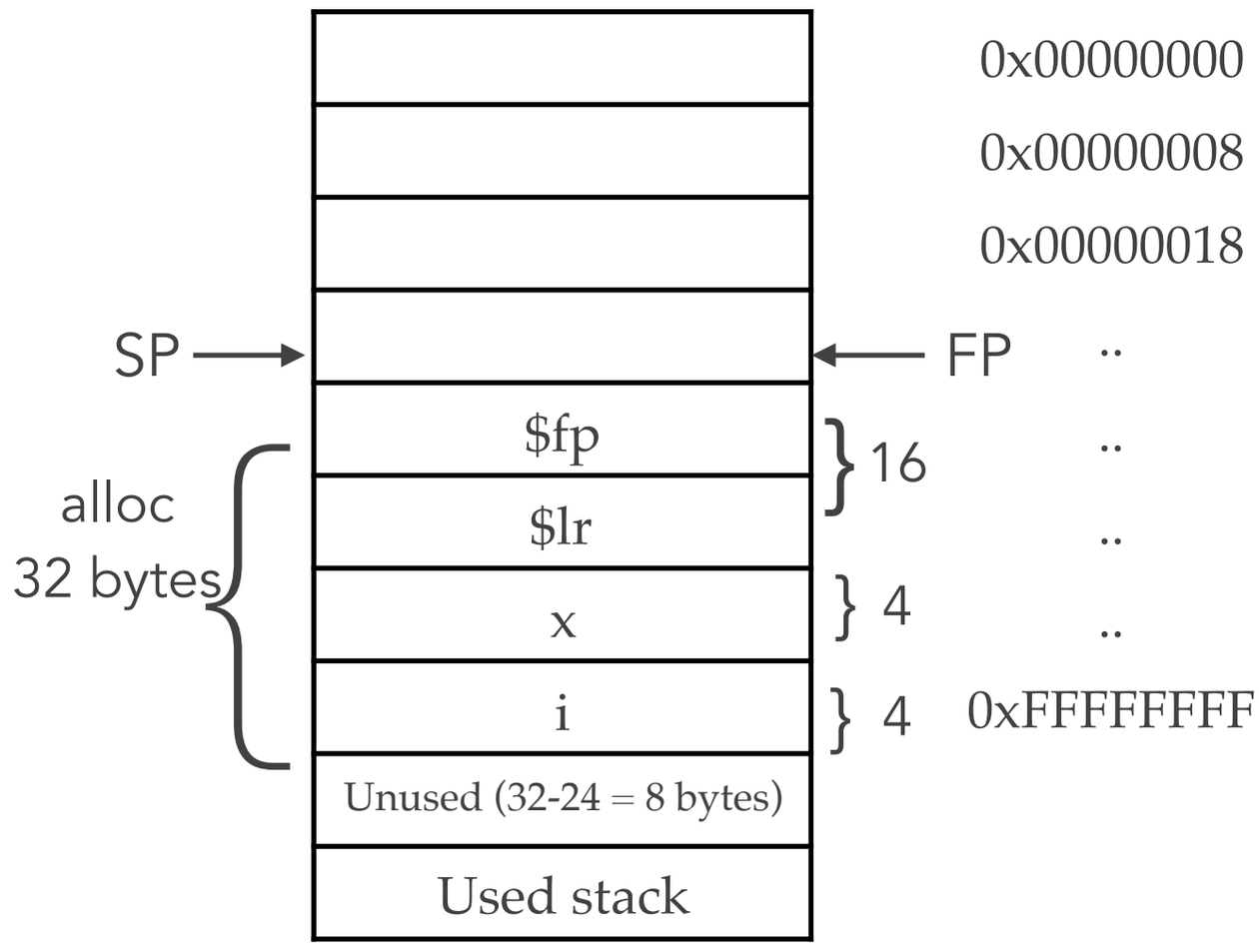
Calculating space required

```
alloc = -(16 + [memory required for local variables]) & -16
dealloc = -alloc
```

alloc is negative, because we need to decrement the SP (Stack uses high memory, and grows towards lower memory addresses).

Using Equates

```
size = 4+4
alloc = -(16 + size) & -16
      = -32
dealloc = -alloc
main:
   stp  fp, lr, [sp, alloc]!
   ldp fp, lr, [sp], dealloc
```



fin.