Lec 14

Thursday, June 9, 2022 5:47 AM

Real World Instruction Sets

Arch	Type	# Oper	# Mem	Data Size	# Regs	Addr Size	Use
Alpha	Reg-Reg	3	0	64-bit	32	64-bit	Workstation
ARM	Reg-Reg	3	0	32/64-bit	16	32/64-bit	Cell Phones, Embedded
MIPS	Reg-Reg	3	0	32/64-bit	32	32/64-bit	Workstation, Embedded
SPARC	Reg-Reg	3	0	32/64-bit	24-32	32/64-bit	Workstation
TI C6000	Reg-Reg	3	0	32-bit	32	32-bit	DSP
IBM 360	Reg-Mem	2	1	32-bit	16	24/31/64	Mainframe
x86	Reg-Mem	2	1	8/16/32/ 64-bit	4/8/24	16/32/64	Personal Computers
VAX	Mem-Mem	3	3	32-bit	16	32-bit	Minicomputer
Mot. 6800	Accum.	1	1/2	8-bit	0	16-bit	Microcontroler

Logical Instn

and r_1, r_2, r_3

$r_2 \rightarrow 1010$

$r_3 \rightarrow 0110$

$r_1 \rightarrow 0010$

$(r_2 \cap r_3)$

Shift Instn

Logical Shift left (lsl)

<<

0010 << 2 → 1000

$<< n$ is same as multiplying by 2^n

Arithmetic Shift right (asr)

>> operator.

0010 >> 1 → 0001

0010 >> 2 → 0000

1000 >> 2 → 1110

same as diving a signed number by 2^n.

Logical shift right (lsr) (>>> operator)

1000 >>> 2 → 0010

Same as diving an unsigned number 2^n.

Compute 102 * 7.5

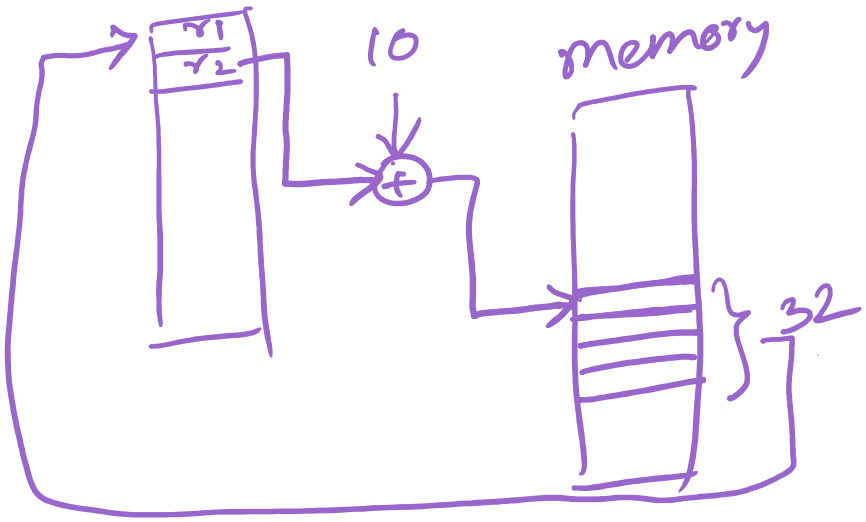
102(8 − 0.5)

```
mov r0, 102
lsl r1, r0, 3
lsr r2, r0, 1
```

sub r_3, r_1, r_2

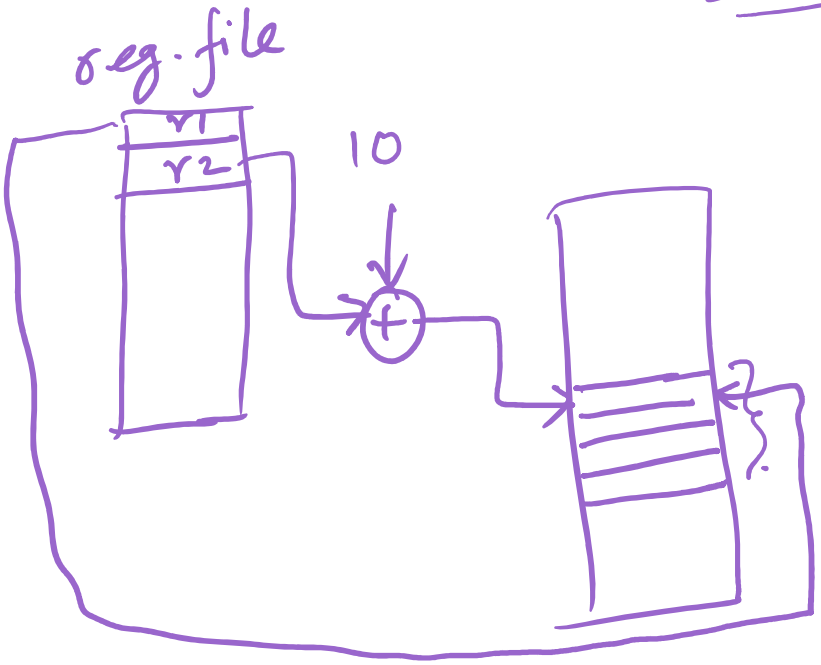
Load - store inst^n

ld $r1, 10[r2]$ $\quad r1 \leftarrow [r2+10]$



St $r1, 10[r2]$

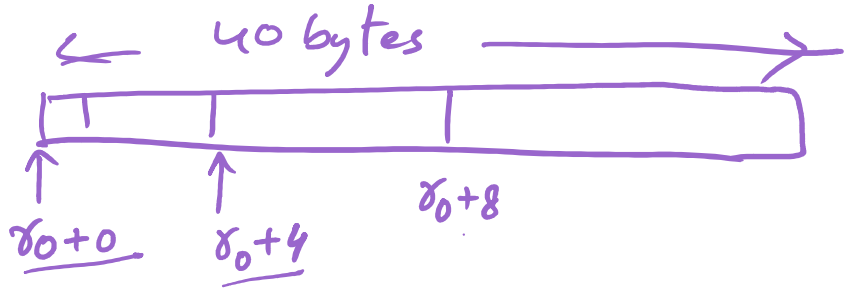
$[r2+10] \leftarrow r1$



Example ← 40 bytes →

Example

```
int arr[10]
```




```
arr[3] = 5 ;
arr[4] = 8 ;
arr[5] = arr[4] + arr[3] ;
```

assume base of array is saved in r_0.

```
mov r1, 5
st r1, 12[r0]
mov r2, 8
st r2, 16[r0]
add r3, r1, r2
st r3, 20[r0].
```

Factorial (num)

```
int prod = 1 ;
int idx ;
for (idx = num; idx > 1; idx --) {
    prod = prod * idx
}.
```

Assembly code

```
mov r1, 1      /* prod = 1 */
mov r2, r0     /* idx = num */
.loop:
mul r1 r1 r2   /* prod = prod * idx */
```

```
mul r1, r1, r2   /* prod = prod * idx */
sub r2, r2, 1    /* idx = idx - 1 */
cmp r2, 1        /* Compare(idx, 1) */
bgt .loop        /* if (idx > 1) goto .loop */
```

Modifiers (for instructions that has an immediate operand)

* default : mov → treat the 16 bit immediate as a signed number (automatic sign extension)

* (u) : movu → treat the 16 bit immediate as unsigned number.

* (h) : movh → Left shift the 16 bit immediate by 16 positions.

```
mov r1, 0XAB12
```


```
movu r1, 0XAB12
```

$r_0 \leftarrow$ 0X12AB A92D

```
movh r0, 0X12AB
```

movu r1, 0XAB12

movh r1, 0XAB12

addu, r0, 0XA92D

Instruction format