Indian Institute of Technology, Kharagpur
Department of Computer Science and Engineering

CS31007 : COMPUTER ORGANISATION AND ARCHITECTURE

ASSIGNMENT

29th October, 2022

Atishay Jain (20CS30008)

Contents

One Instruction Set CPU

1. Instruction:

We have to implement the One Instruction Set CPU. We choose the "Subtract and branch on less than or equal to zero" (**SUBLEQ**) instruction and implement the architecture design accordingly.

Instruction: *subleq* a, b, c

1.1 SUBLEQ:

The subleq instruction performs the following operations:

- Subtracts the content stored in memory address pointed to by a and pointed by b and store the result in address pointed by b.

$$Mem[b] = Mem[b] - Mem[a]$$

- If the result of the subtraction is less than or equal to 0, the control will be transferred to address pointed by c, else the control is transferred to the next instruction in the sequence.

$$if(Mem[b] \leq 0)$$
$$goto\ c$$

- If we do not pass the third operand(i.e. c), then the control will always be passed to the next instruction in the sequence after subtraction.

1.2 Synthesised Instruction:

We can deduce other instructions using this single instruction.

1.2.1 Jump

Jump to c

```
subleq a, a, c
```

1.2.2 Add

Add the content of memory address pointed by a and b and store the result in memory address pointed by b.

- We will be using a temporary memory address t(initially storing 0) to store -a.
  $Mem[t] = Mem[t] - Mem[a] = -Mem[a]$
- Then we will subtract -a from b.
  $Mem[b] = Mem[b] - (-Mem[a])$
- We then set back the content of t to 0 again.
  $Mem[t] = Mem[t] - Mem[t] = 0$

```
subleq a, t
subleq t, b
subleq t, t
```

1.2.3 Sub

Subtract the content of memory address pointed by a and b and store the result in memory address pointed by b.

```
subleq a, b
```

1.2.4 Copy

Copy the content of memory address pointed by a in the memory address pointed by b.

- First, set the content of b to 0.
  $Mem[b] = Mem[b] - Mem[b] = 0$
- We will use a temporary variable t(initially containing 0), to store -a.
  $Mem[t] = Mem[t] - Mem[a] = -Mem[a]$
- Now, we subtract t(= -a) from b(=0) and store the result in b.
  $Mem[b] = Mem[b] - Mem[t]$
- Then, set the content of t again to 0.
  $Mem[t] = Mem[t] - Mem[t] = 0$

```
subleq b, b
subleq a, t
subleq t, b
subleq t, t
```

2. Multiplication

Instruction: mul a, b
$Mem[b] = Mem[a] * Mem[b]$

We will code the multiplication operation using the single instruction that we chose earlier($SUBLEQ$). Let us start with the high level instruction set need for multiplication operation.

- Run a loop which will add 'a' to the 'result' and decrease 'b' by 1.
- Continue this loop until 'b' is greater than 0.
- We will using two labels to the memory addresses to implement the loop.
- We will be using three temporary addresses t1(used as iterator), t2(storing -a) and t3(used for uconditional jump.
- We will be assuming that there is some memory address(X) whose content is set to 1 and some memory address(Y) whose content is set to 0 and some memory address(Z) whose content is set to -1.

$Mem[t1] = Mem[b]$
$Mem[t2] = -Mem[a]$
$Mem[b] = 0$
$if(Mem[t1] <= 0)$ $goto$ $loop2$

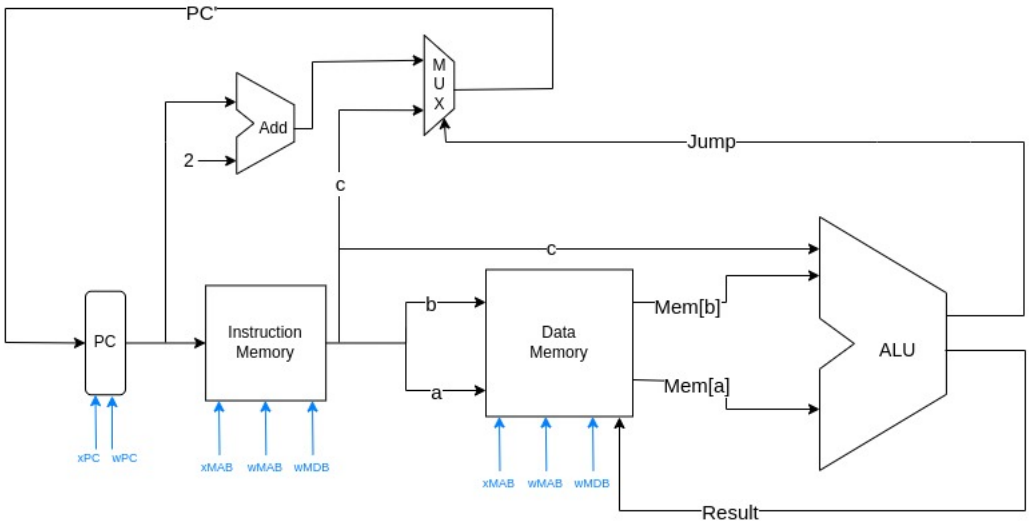
$loop1$:
$if(Mem[t1] <= 0)$ $goto$ end
$Mem[b] = Mem[b] - Mem[t2]$
$Mem[t1] = Mem[t1] - 1$
$goto$ $loop1$

$loop2$:
$if(-Mem[t1] <= 0)$ $goto$ end
$Mem[b] = Mem[b] - Mem[t2]$
$Mem[t1] = Mem[t1] + 1$
$goto$ $loop2$

end :

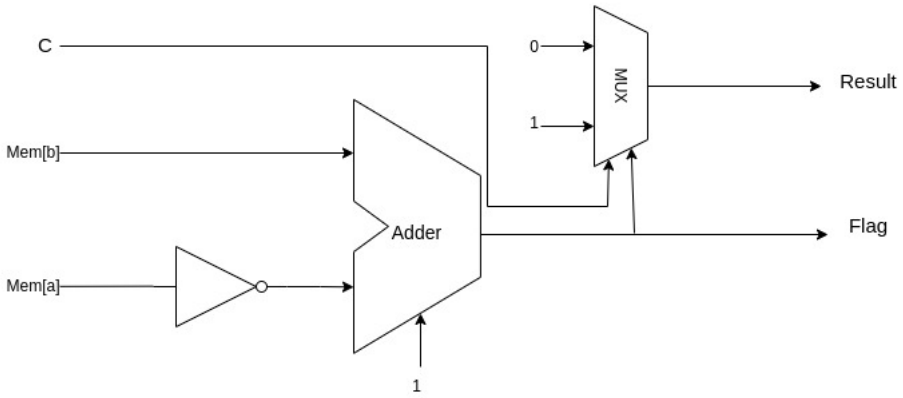
```
    subleq t1, t1
    subleq t2, t2
    subleq t3, t3
    subleq b, t2
    subleq t2, t1
    subleq t2, t2
    subleq a, t2
    subleq b, b
    subleq Y, t1, loop2
loop1:
    subleq Y, t1, end
    subleq t2, b
    subleq X, t1
    subleq t3, t3, loop
loop2:
    subleq t1, t3, end
    subleq t2, b
    subleq Z, t1
    subleq t3, t3, loop2
end:
```

3. Datapath



NOTE: The MAB and MDB components are not shown here for simplicity.

3.1 Arithmetic Logical Unit



3.2 Instruction

3.3 Controller Requirements:

3.3.1 Instruction Fetching:

MAB $\leftarrow PC[xMAB \leftarrow 11, wMAB \leftarrow 1]$
$MDB \leftarrow MEM[wMDB \leftarrow 1]$

3.3.2 Data Fetching:

Fetch Mem[a]:
MAB $\leftarrow a[xMAB \leftarrow 01, wMAB \leftarrow 1]$
$MDB \leftarrow MEM[wMDB \leftarrow 1]$

$FetchMem[b]$:
$MAB \leftarrow [xMAB \leftarrow 1]$
$MDB \leftarrow MEM[wMDBj \leftarrow 1]$

3.3.3 Inside ALU:

Result = Mem[b] - Mem[a]
For the Multiplexer inside ALU:
if(c == 0 or Result > 0)
Jump $\leftarrow 0$
else
$Jump \leftarrow 1$

3.3.4 Multiplexer Outside ALU:

if(Jump)
PC' $\leftarrow PC + 2$
else

$PC' \leftarrow c$

3.3.5 Others:

Mem[b] $\leftarrow Result$
Controls used for this:
MAB $\leftarrow b[xMAB \leftarrow 11, wMAB \leftarrow 1]$

MDB $\leftarrow Result[wMDB \leftarrow 0(tobereadfromALU)]$

$Afterthis$,
$PC \leftarrow PC'[xPC \leftarrow 11, wPC \leftarrow 1]$