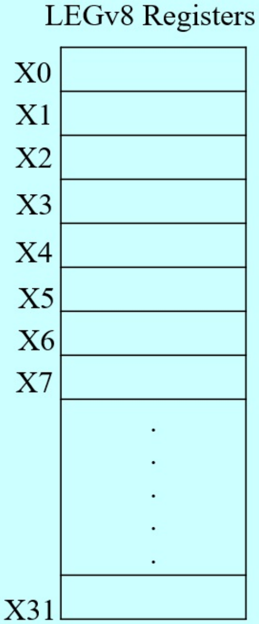
Software simulation
ARMSim

Assembly language will be simulated using ARMSim Simulator. LEGv8 and ARMSim (ARMv7) are almost similar with little distinction.

→ * ARMSim (windows)

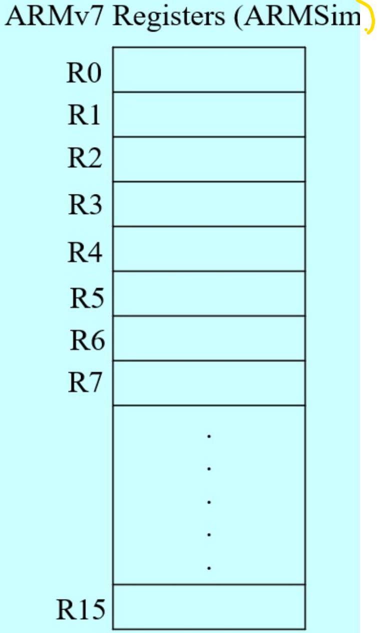
* Online editor

LEGv8 VS ARMv7 (ARMSim) Registers



LEGv8 Registers

X0
X1
X2
X3
X4
X5
X6
X7
X31

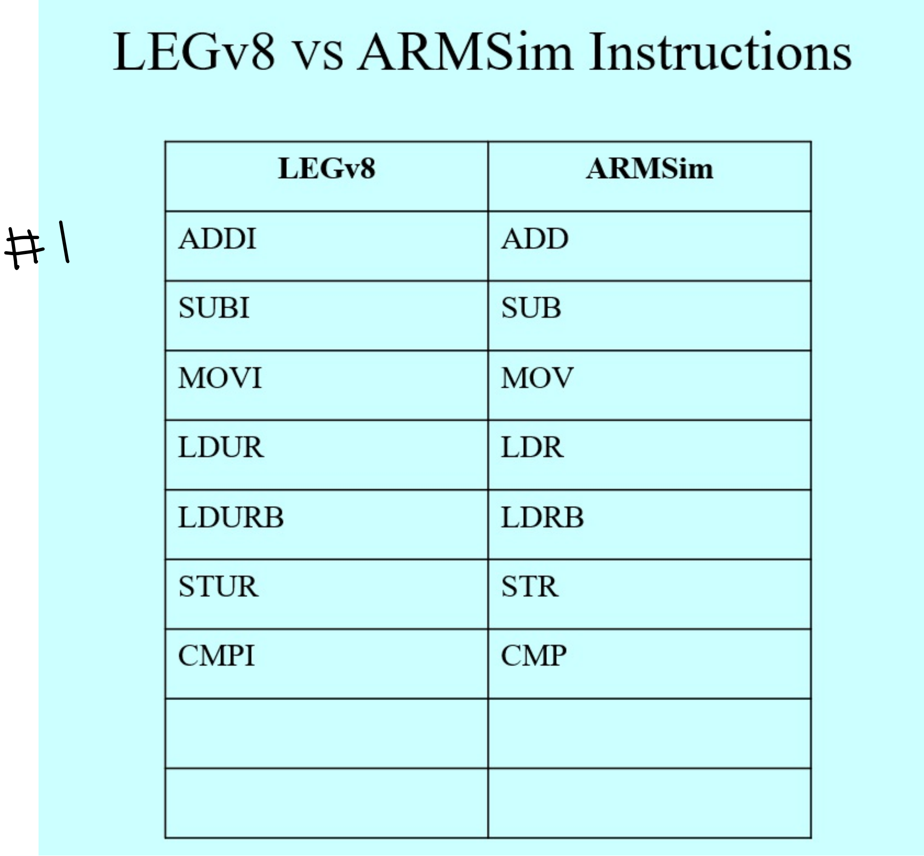


ARMv7 Registers (ARMSim)

R0
R1
R2
R3
R4
R5
R6
R7
R15

ADD X1, X2, X3 → ADD R1, R2, R3

ADDI X9, X9, #1



LEGv8 vs ARMSim Instructions

LEGv8	ARMSim
ADDI	ADD
SUBI	SUB
MOVI	MOV
LDUR	LDR
LDURB	LDRB
STUR	STR
CMPI	CMP

ADD R9, R9, #1

Program: A C++ program adds two given numbers a=10 and b=20 and prints resulting number. Convert the following C++ code to assembly code and simulate using ARMSim Version 2.1.

X2 → a   X3 → b
X1 → c

c++ code

```
#include<iostream>
using namespace std;
int main()
{
   int a=10, b=20, c;
   c= a + b;      c = 10+20
   cout<<c;         = 30
   return 0;
}
```

output

```
30
```

LEGv8 Instructions:

```
MOVI X2, #10        // X2 = 10
MOVI  X3, #20       // X3 = 20
ADD X1, X2, X3      // X1 = 10+20 = 30
```

ARMSim Instructions:

```
MOV R2, #10         // R2 = 10
MOV R3, #20         // R3 = 20
ADD R1, R2, R3      // R1 = 10 +20 = 30
```

Program: A C++ program to find the factorial of 3. Convert the following C++ code to assembly code and simulate using ARMSim version 2.1.

$3! = 3 \cdot 2 \cdot 1 = 6$

C++ Code:

```
#include <iostream>
using namespace std;

int main()
{
  int i, j=1, n=3;
  for (i = 1; i <= n; i++)
  {
     j = j * i;
  }
  cout<<j;
}
```

Output

$\boxed{6}$

1 <= 3 → True

2 <= 3 → True

3 <= 3 → True

4 <= 3 → False

n	i	J
3	1	1
	2	2
	3	6
	4	

$J = 1 * 1 = 1$

$J = 1 * 2 = 2$

$J = 2 * 3 = 6$

Program: A C++ program to find the factorial of 3. Convert the following C++ code to assembly code and simulate using ARMSim version 2.1.

R2 → n
R5 → j
R6 → i

ARMSim Instructions:

```
        MOV R2, #3
        MOV R5, #1
        MOV R6, #1
loop:   CMP R6, R2
        BGT exit
        MUL R5, R5, R6
        ADD R6, R6, #1
        B loop
exit:
```

// n = 3
// j = 1
// i = 1
// compare i and n
// j = j * i
// i = i+1

i > n
1 > 3 → False
2 > 3 → False
3 > 3 → False
4 > 3 → True

	(j)	(i)
R2	R5	R6
3	1	1
	2	2
	$\boxed{6}$	3
		4

R5 = 1 * 1 = 1
R5 = 1 * 2 = 2
R5 = 2 * 3 = 6

R5 = 6