Assignment 3 "Condition Codes and Jumps"

Group - SS05

The CPU is based on MIPS architecture.

- **Memory:** 2048 locations of 1 bytes each (mem[2048]) = 2KB of memory

Memory	Memory Address	Address in program	Locations (bytes)
Data memory	0x0481 to 0x07FF	mem[1053] to mem[2047]	896
Stack	0x0400 to 0x0480	mem[1024] to mem[1152]	128
Instruction memory	0x0200 to 0x03FF	mem[512] to mem[1023]	512
OS memory	0x0000 to 0x01FF	mem[0] to mem[511]	512

- **Instruction size:** 4 bytes = 32 bits

Opcode (8 bits)	Operand 1 (8 bits)	Operand 2 (8 bits)	Operand 3 (8 bits)

- **Opcodes:**

Register	Opcode	Instruction	Opcode
r0	0x00	lw	0x00
r1	0x01	sw	0x01
r2	0x02	add	0x02
r3	0x03	sub	0x03
r4	0x04	mul	0x04
r5	0x05	div	0x05
r6	0x06	mod	0x06
r7	0x07	push	0x07
r8	0x08	pop	0x08
r9	0x09	lea	0x09
r10	0x0A	beq	0x0A
r11	0x0B	bne	0x0B
r12	0x0C	slt	0x0C
r13	0x0D	j	0x0D
r14	0x0E	jr	0x0E
r15	0x0F	mov	0x0F
		mvi	0x10
		inc	0x11
		dec	0x12

- **Status register:** 8bits

-	-	OF	SF	ZF	AC	PF	CF
(Bit 7)	(Bit 6)	(Bit 5)	(Bit 4)	(Bit 3)	(Bit 2)	(Bit 1)	(Bit 0)

Instructions representation:

1. **lea r0,r1,r2**

   Big Endian machine

lea (0x09)	r0 (0x00)	r1 (0x01)	r2 (0x02)

2. **beq r0,r1,lb0**

   Big Endian machine

beq (0x0A)	r0 (0x00)	r1 (0x01)	lb(line number)

3. **bne r0,r1,lb0**

   Big Endian machine

bne (0x0B)	r0 (0x00)	r1 (0x01)	lb(line number)

4. **slt r0,r1,r2**

   Big Endian machine

slt (0x0C)	r0 (0x00)	r1 (0x01)	r2 (0x02)

5. **j lb0**

   Big Endian machine

j (0x0D)	lb(line number)	0 (0x00)	0 (0x00)

6. **jr r0**

   Big Endian machine

jr (0x0E)	r0 (0x00)	0 (0x00)	0 (0x00)

Loops implemented

NOTE: In assembly.txt we have kept code for do-while loop only.

For demo of other loops please copy the code from while_loop.txt, for_loop.txt and paste it in assembly.txt and execute the program.

1. **Do-while loop**

   We implemented a program which finds factorial of the number stored in r1.

C	**Assembly**

int a= 5
int result=a;
do
{
a--;
result=result*a;
}
while(a>1)
	mvi r1,0005
	mvi r3,0001
	mov r5,r1
	lb0:dec r1
	mul r5,r5,r1
	bne r1,r3,lb0

- **CPU State initially**

```
Machine selection 1.Little Endian 2. Big Endian : 1

********** Cpu state initially **********

Content of instruction Memory:

Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value
200     ff    201     ff    202     ff    203     ff    204     ff    205     ff
206     ff    207     ff    208     ff    209     ff    20a     ff    20b     ff
20c     ff    20d     ff    20e     ff    20f     ff    210     ff    211     ff
212     ff    213     ff    214     ff    215     ff    216     ff    217     ff
218     ff    219     ff    21a     ff    21b     ff    21c     ff    21d     ff
21e     ff    21f     ff

Content of Stack Memory:

Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value
400     ff    401     ff    402     ff    403     ff    404     ff    405     ff
406     ff    407     ff    408     ff    409     ff    40A     ff    40B     ff
40C     ff    40D     ff    40E     ff    40F     ff    410     ff    411     ff
412     ff    413     ff    414     ff    415     ff    416     ff    417     ff
418     ff    419     ff    41A     ff    41B     ff    41C     ff    41D     ff
41E     ff    41F     ff

Content of Data Memory:

Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value
481     ff    482     ff    483     ff    484     ff    485     ff    486     ff
487     ff    488     ff    489     ff    48A     ff    48B     ff    48C     ff
48D     ff    48E     ff    48F     ff    490     ff    491     ff    492     ff
493     ff    494     ff    495     ff    496     ff    497     ff    498     ff
499     ff    49A     ff    49B     ff    49C     ff    49D     ff    49E     ff
49F     ff    4A0     ff

Content of General Purpose Register:

R0 = 00000000 R1 = 00000000  R2 = 00000000  R3 = 00000000  R4 = 00000000  R5 = 00000000  R6 = 00000000  R7 = 00000000
R8 = 00000000 R9 = 00000000 R10 = 00000000 R11 = 00000000 R12 = 00000000 R13 = 00000000 R14 = 00000000 R15 = 00000000

Contents of Status register:    0 0 0 0 0 0 0 0

Content of Program Counter (PC): 00000000

Content of Stack Pointer (SP):  00000400

Content of Base Pointer (BP):   00000400
```

- **CPU State after loading instructions in memory (Big Endian Machine)**

```
Machine selection 1.Little Endian 2. Big Endian : 2

********** Cpu state after loading instructions into memory **********

Content of instruction Memory:

Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value
200     10    201     01    202     00    203     05    204     10    205     03
206     00    207     01    208     0f    209     05    20a     01    20b     00
20c     12    20d     01    20e     00    20f     00    210     04    211     05
212     05    213     01    214     0b    215     01    216     03    217     03
218     ff    219     ff    21a     ff    21b     ff    21c     ff    21d     ff
21e     ff    21f     ff    220     ff    221     ff    222     ff    223     ff
224     ff    225     ff    226     ff    227     ff    228     ff    229     ff
22a     ff    22b     ff    22c     ff    22d     ff    22e     ff    22f     ff
```

- **After loading r1 and r3**

```
Content of General Purpose Register:

R0 = 00000000 R1 = 00000005  R2 = 00000000  R3 = 00000001  R4 = 00000000  R5 = 00000000
R8 = 00000000 R9 = 00000000 R10 = 00000000 R11 = 00000000 R12 = 00000000 R13 = 00000000
```

- **After execution of whole program (r5 contains factorial of original number present in r1)**
  (We get output after 17 instruction executions)

```
Content of General Purpose Register:

R0 = 00000000 R1 = 00000001  R2 = 00000000  R3 = 00000001  R4 = 00000000  R5 = 00000078
R8 = 00000000 R9 = 00000000 R10 = 00000000 R11 = 00000000 R12 = 00000000 R13 = 00000000
```

2. While loop

We implemented a program which finds factorial of the number stored in r1.

C	Assembly

int a= 4
int result=a;
while(a>1)
{
  a--;
  result=result*a;
}
	mvi r1,0004
	mvi r3,0001
	mov r5,r1
	lb0:beq r1,r3,lb1
	dec r1
	mul r5,r5,r1
	j lb0
	lb1:

- **CPU State after loading instructions in memory (Big Endian Machine)**

```
Machine selection 1.Little Endian 2. Big Endian : 2
```

```
*************** Cpu state after loading instructions into memory ***************

Content of instruction Memory:

Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value
200     10   201     01   202     00   203     04   204     10   205     03
206     00   207     01   208     0f   209     05   20a     01   20b     00
20c     0a   20d     01   20e     03   20f     07   210     12   211     01
212     00   213     00   214     04   215     05   216     05   217     01
218     0d   219     03   21a     00   21b     00   21c     ff   21d     ff
21e     ff   21f     ff   220     ff   221     ff   222     ff   223     ff
224     ff   225     ff   226     ff   227     ff   228     ff   229     ff
22a     ff   22b     ff   22c     ff   22d     ff   22e     ff   22f     ff
```

- **After loading r1 and r3**

```
Content of General Purpose Register:

R0 = 00000000 R1 = 00000004  R2 = 00000000  R3 = 00000001  R4 = 00000000  R5 = 00000000
R8 = 00000000 R9 = 00000000 R10 = 00000000 R11 = 00000000 R12 = 00000000 R13 = 00000000
```

- **After execution of whole program (r5 contains factorial of original number present in r1)**
  (We get output after 16 instruction executions)

```
Content of General Purpose Register:

R0 = 00000000 R1 = 00000001  R2 = 00000000  R3 = 00000001  R4 = 00000000  R5 = 00000018
R8 = 00000000 R9 = 00000000 R10 = 00000000 R11 = 00000000 R12 = 00000000 R13 = 00000000
```

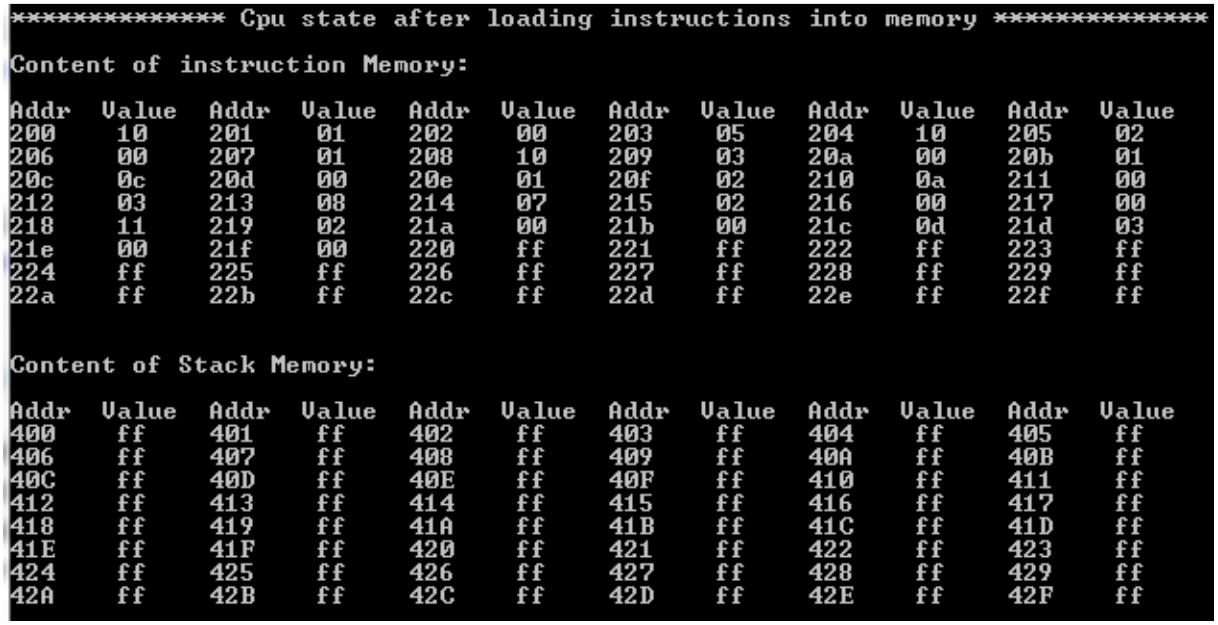
3. For loop

We implemented a program which pushes the number from 1 to r1 to the stack

C	Assembly

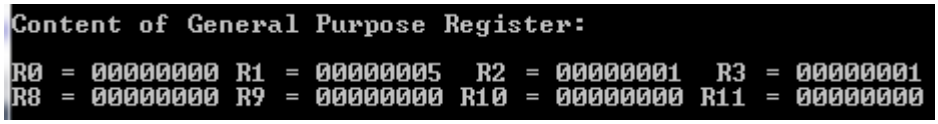
int c;
for (c=1;c<=5;c++)
{
  push(c);
}
	mvi r1,0005
	mvi r2,0001
	mvi r3,0001
	lb2:slt r0,r1,r2
	beq r0,r3,lb1
	push r2
	inc r2
	j lb2
	lb1:

- **CPU State after loading instructions in memory (Big Endian Machine)**



```
*************** Cpu state after loading instructions into memory ***************

Content of instruction Memory:

Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value
200     10   201     01   202     00   203     05   204     10   205     02
206     00   207     01   208     10   209     03   20a     00   20b     01
20c     0c   20d     00   20e     01   20f     02   210     0a   211     00
212     03   213     08   214     07   215     02   216     00   217     00
218     11   219     02   21a     00   21b     00   21c     0d   21d     03
21e     00   21f     00   220     ff   221     ff   222     ff   223     ff
224     ff   225     ff   226     ff   227     ff   228     ff   229     ff
22a     ff   22b     ff   22c     ff   22d     ff   22e     ff   22f     ff


Content of Stack Memory:

Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value
400     ff   401     ff   402     ff   403     ff   404     ff   405     ff
406     ff   407     ff   408     ff   409     ff   40A     ff   40B     ff
40C     ff   40D     ff   40E     ff   40F     ff   410     ff   411     ff
412     ff   413     ff   414     ff   415     ff   416     ff   417     ff
418     ff   419     ff   41A     ff   41B     ff   41C     ff   41D     ff
41E     ff   41F     ff   420     ff   421     ff   422     ff   423     ff
424     ff   425     ff   426     ff   427     ff   428     ff   429     ff
42A     ff   42B     ff   42C     ff   42D     ff   42E     ff   42F     ff
```

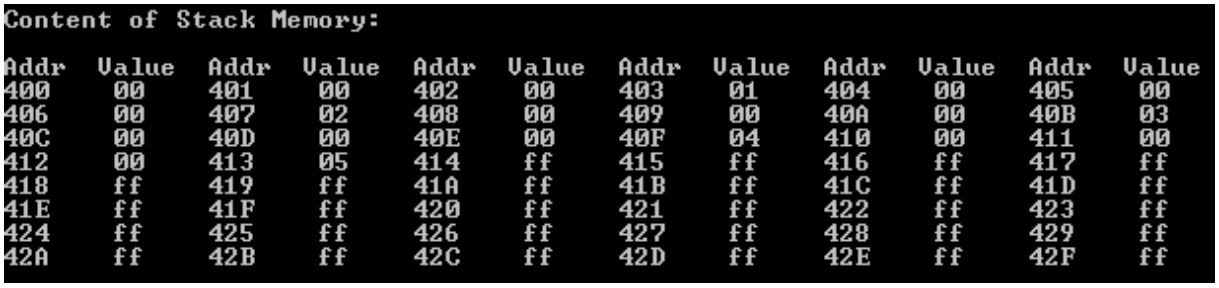
- **After loading values of r1, r2, r3**



```
Content of General Purpose Register:

R0 = 00000000 R1 = 00000005  R2 = 00000001  R3 = 00000001
R8 = 00000000 R9 = 00000000 R10 = 00000000 R11 = 00000000
```

- **After execution of whole program**

  (We get output after 26 instruction executions)



```
Content of Stack Memory:

Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value  Addr  Value
400     00   401     00   402     00   403     01   404     00   405     00
406     00   407     02   408     00   409     00   40A     00   40B     03
40C     00   40D     00   40E     00   40F     04   410     00   411     00
412     00   413     05   414     ff   415     ff   416     ff   417     ff
418     ff   419     ff   41A     ff   41B     ff   41C     ff   41D     ff
41E     ff   41F     ff   420     ff   421     ff   422     ff   423     ff
424     ff   425     ff   426     ff   427     ff   428     ff   429     ff
42A     ff   42B     ff   42C     ff   42D     ff   42E     ff   42F     ff
```