

1

Considering the following code in a GPU kernel function:

```
int index = blockIdx.x * blockDim.x + threadIdx.x;

if (index > x1)      // Section 1 - THEN
{
    ...
}

else                // Section 1 - ELSE
{
    ...
}

d_e[index] = d_c[index] + d_d[index];
```

Assuming that the corresponding assembly code is as follows:

PC	Instruction	Comment
0000	MOV R1, c [0x1] [0x100];	
0020	SSY 0xf0;	SSY Instruction (push stack)
0028
0090	@P0 BRA 0xb8;	Branch corresponding to Outer IF Then-Else (push stack if divergent)
0098	LD.E R3, [R2];	ELSE PART
...		
00b0	ST.E.S [R6], R2;	ELSE PART. Notice ".S" flag. (pop stack)
00b8	LD.E R5, [R4];	IF THEN PART
...
00e8	NOP.S CC.T;	Last Instruction of Outer If-Then. Notice ".S" flag. (pop stack)
00f0	LD.E R3, [R10];	Threads Synchronizes at this point.
00f8	LD.E R4, [R8];	...
0100	LD.E R2, [R6];	Go till Exit.
...
0140	EXIT;	

With a warp size of 8, show how the SIMT stack is updated during the execution for the following two warps. (1) The branch at PC=x0090 is taken for all 8 threads in a warp. (2) Among the 8 threads in a warp, the branch outcomes are: taken for the first 2 threads and not-taken for the remaining 6 threads. (Hint: you can refer to the slides in our course notes to see one format that we used to show the SIMT stack states)

Case (1): Branch at PC=x0090 is taken for all 8 threads in a warp (no divergence).

PC	Active Mask	TOS	TOS-1	TOS-2	Comments
0000	11111111				
0020	11111111				Before SSY
0028	11111111	SSY 0xf0			After SSY
0090	11111111	SSY 0xf0			Before Branch
00b8	11111111	SSY 0xf0			After Branch
...	11111111	SSY 0xf0			
00e8	11111111	SSY 0xf0			NOP.S (pop)
00f0	11111111				LD.E
00f8	11111111				LD.E
0100	11111111				LD.E
...	11111111				
0140	11111111				EXIT

Case (2): Branch at PC=x0090 is taken for first two threads and not taken for other 6 (divergence)

PC	Active Mask	TOS	TOS-1	TOS-2	Comments
0000	11111111				
0020	11111111				Before SSY
0028	11111111	SSY 0xf0			After SSY
0090	00111111	SSY 0xf0			Before Branch
0098	00111111	DIV 0xf0 0xb8	SSY 0xf0		After Branch
...	00111111	DIV 0xf0 0xb8	SSY 0xf0		
00b0	00111111	DIV 0xf0 0xb8	SSY 0xf0		Before ST.E.S (pop)
00b8	11000000	SSY 0xf0			After ST.E.S
...	11000000	SSY 0xf0			
00e8	11000000	SSY 0xf0			NOP.S (pop)
00f0	11111111				LD.E
00f8	11111111				LD.E
0100	11111111				LD.E
...	11111111				
0140	11111111				EXIT