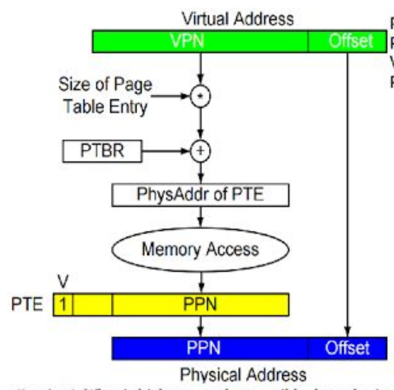


PTBR	page table base register
PTE	page table entry
VPN	virtual page number
PPN	physical page number

(3 points) What is highest speedup possible through pipelining for a 6 instruction program if latch delay is 2 ns and total combinational logic delay of a non-pipelined design is 10 ns?

The highest speedup can be obtained when dividing the total combinational logic into N stages, which each stage taking $10/N$ ns.

On the non-pipelined processor, the program takes $6 \times 10 = 60$ ns to run.

On a pipelined processor, each stage takes $\frac{10}{N} + 2$ ns. The program in total takes

$$
\begin{aligned}
&(N-1+6) \times \left(\frac{10}{N}+2\right) \\
&= (N+5)\left(\frac{10}{N}+2\right) \text{ns}
\end{aligned}
$$

The total time is minimized for $N = 5$. The speedup is $\frac{60}{40} = 1.5$.

(12 points) Fill in the blanks and calculate the cache access times for the following actions in sequence (the second action follows immediately after the first one). Show your calculations for full credit. Write the address in hex and circle hit or miss.

i. Read virtual address 4x00107

- Cache access time: 207 ns
- Physical address: x3307
- Cache hit / [✓ miss]
- TLB hit / [✓ miss]

```
VA: 00 0000 0001 0000 0111  TLB miss on VPN = 00 0000 0001
PA:    0011 0011 0000 0111  Cache Compulsory Miss
```

$5+80+5=90$ ns to get the physical tag
4 ns to detect a cache miss by tag comparison
100 ns to access the main memory
$10+3=13$ ns to access data (tag access can be done in parallel)

ii. Read virtual address 4x34500

- Cache access time: 13 ns
- Physical address: x3300
- Cache [✓ hit] / miss
- TLB [✓ hit] / miss

```
VA: 11 0100 0101 0000 0000  TLB hit on VPN = 11 0100 0101
PA:    0011 0011 0000 0000  Cache hit
```

Parallelism:
Indexing tag + tag comparison $= 8+4 = 12$ ns
Indexing data + multiplexing data $= 10+3 = 13$ ns

- Byte addressable
- 256 byte (2^8 byte) page size

Cache:

- Virtually-indexed and physically-tagged
- 4-way set-associative with ~~6~~ 4 index bits
- ~~4 KB ($2^{1}2$ byte)~~ 1 KB ($2^{1}0$ byte) data storage (excluding bits for dirty, valid, tag and LRU)
- Read allocate policy
- Indexing the data array takes 10 ns
- Indexing the tag array takes 8 ns
- Tag comparison takes 4 ns
- Multiplexing the output data takes 3 ns
- A cache miss takes 100 ns to access the main memory and allocate to the cache line
- Assume a hit or miss is detected immediately after the tag comparison
- Initially empty (all lines are invalid)

TLB:

- Fully-associative
- A TLB access takes 5 ns
- Read allocate policy
- TLB is updated on a TLB miss
- All entries are listed below

Valid	VPN	PPN
0	00 0000 0001	0001 0000
1	00 0000 0010	0000 1110
0	00 0001 0110	0011 0011
0	00 0001 1011	0000 0000
1	11 1010 0100	1000 0100
1	11 0100 0101	0011 0011
1	10 0010 1010	1100 0110
0	00 0000 0000	0000 0001

Page Table:

- Single level page table
- A page table access takes 80 ns
- Some of the entries are listed below

Valid	VPN	PPN
0	00 0000 0000	0001 0000
1	00 0000 0001	0011 0011
1	00 1101 0010	0000 0000
0	01 0000 0011	0010 0100
1	01 1010 0010	1110 0001
1	01 1111 1101	0000 1110
1	10 0000 0110	1100 0110
0	11 1111 1111	0101 0110

- Fast cache access
  - Only require address translation upon miss
- Issues
  - Homonym
    - Same VA maps to different PAs upon context switch
  - Synonym (also a problem in VIPT)
    - Different VAs map to the same PA when data is shared by multiple processes

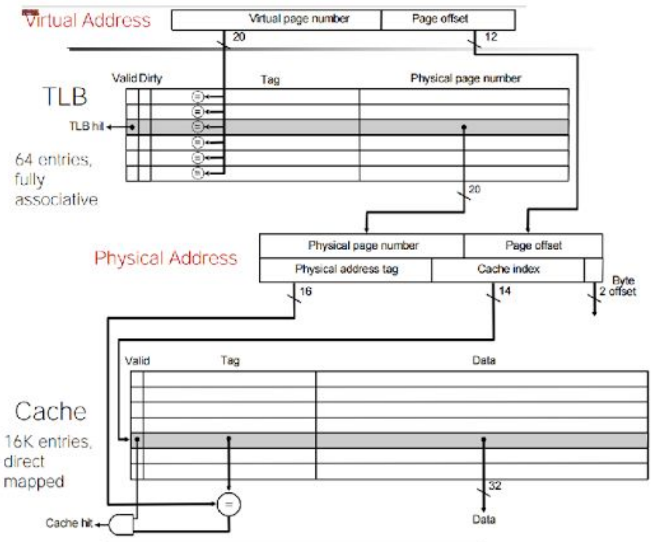
```
add $t0, $s0, $s1
xor $t1, $t0, $s2
lw  $s0, -12($a0)
sub $s5, $s0, $s1
```

ssible to resolve any of th
rwarding would be unnece

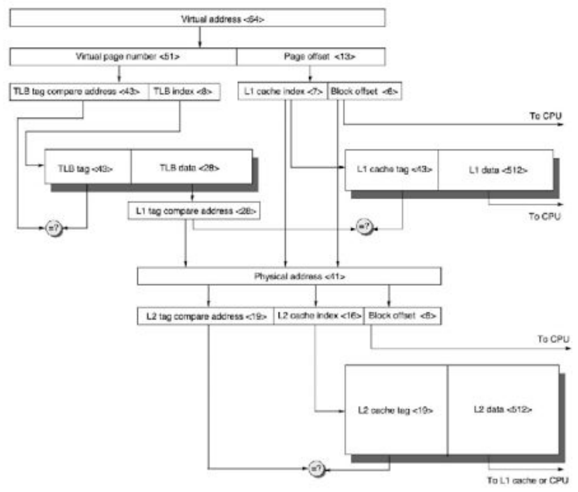
ssible to resolve one, but r
e sub.

```
add $t0, $s0, $s1
lw  $s0, -12($a0)
sub $s5, $s0, $s1
xor $t1, $t0, $s2
```

- Gain benefit of a VIVT and PIPT
  - Very common in commercial processors
  - Parallel Access to TLB and VIPT cache
- Issues
  - Synonym as VIVT; no homonym



Virtual Address
Virtual page number
Page offset
20
12
TLB
Valid Dirty
Tag
Physical page number
TLB hit
64 entries, fully associative
20
Physical Address
Physical page number
Page offset
Physical address tag
Cache index
Byte offset
16
14
2
Valid
Tag
Data
Cache
16K entries, direct mapped
32
Data
Cache hit



Virtual address <64>
Virtual page number <51>
Page offset <13>
TLB tag compare address <43>
TLB index <8>
L1 cache index <7>
Block offset <6>
To CPU
TLB tag <43>
TLB data <28>
L1 cache tag <43>
L1 data <512>
L1 tag compare address <28>
=?
=?
To CPU
Physical address <41>
L2 tag compare address <19>
L2 cache index <16>
Block offset <6>
To CPU
L2 cache tag <19>
L2 data <512>
=?
To L1 cache or CPU