> A memory system has 1M words. Each word is an 8-bit byte. The memory is organized into blocks of 8 words each. The cache has 256K words, organized into cache lines of 8 words each. The memory cache is organized into 4-way set associative cache. Which of the following statements are true?

```
(a) 13 bits are needed for cache set address.
```

Ans: 快取(cache)有256K個 words，並且是4-way set associative cache。每個快取線(cache line)有8個 words。這意味著有256K/8 = 32K個快取線，分為4路，因此會得到8K為1組cache set address。要地址8K組所需的位數是log2(8K) = 13 bits

```
(b) 4 bits are for Tag.
```

Ans: Tag 用於區分可以映射到同一組的不同區塊(block)。在這種情況下，內存大小為1M words，每個區塊是8個 words，我們有1M/8 = 128K個區塊。
同時，我們有8K組，每個區塊映射到8K組的其中一個。每個組的唯一區塊數 = 128K區塊 / 8K組 = 每組16個區塊。然而，每個組只能容納4個區塊（4-way set associative cache），所以每個組的其他12個區塊需要使用標籤來區分。
在這種情況下，標籤需要對16個blocks進行位址，這需要log2(16) = 4 bits。

```
(c) 21 bits are needed to address all words.
```

Ans: 內存中的words總數為1M。要位址1M的位置，我們需要log2(1M) = 20 bits，不是21 (1M ~= 2^20)

```
(d) 17 bits are needed to address all blocks.
```

Ans: blocks of 8 words each, 所以，內存中的區塊(block)總數是1M / 8 = 128K。要位址128K的位置，我們需要log2(128K) = 17 bits

> (e) None of the above.

> For a typical disk without considering queuing delay, the average seek time is 5 milliseconds, and the transfer rate is 2M-bytes per second. The disk rotates at 600 RPM (Revolution Per Minute), and the controller overhead is 0.5 millisecond. Determine the average time to read a 1792-bytes sector.

**Answer:**

1. Seek Time: This is the time it takes for the read/write head to move to the track where the data is located. It is given as 5 milliseconds.

2. Rotational Delay: This is the time it takes for the disk to rotate until the desired sector is under the read/write head. On average, it takes half a rotation for the desired sector to be accessible. Given that the disk rotates at 600 RPM (Revolutions Per Minute), the time for one full rotation is (1 minute / 600) = 0.1 seconds = 100 milliseconds. Therefore, the average rotational delay (half a rotation) is 100/2 = 50 milliseconds

3. Transfer Time: This is the time it takes to actually read the data. Given that the sector size is 1792 bytes and the transfer rate is 2M bytes per second, the transfer time can be computed as (1792 bytes / 2M bytes/second) = 0.000896 seconds = 0.896 milliseconds.

4. Controller Overhead: This is the time spent on tasks like error checking and correction, which is given as 0.5 milliseconds.

Total Time = Seek Time + Rotational Delay + Transfer Time + Controller Overhead
= 5 ms + 50 ms + 0.896 ms + 0.5 ms
= 56.396 milliseconds

> Assume that the miss rate of an instruction cache is 3% and the miss rate of the data cache is 6%. If a processor has a CPI of 4 without any memory stalls and the miss penalty is 100 cycles for all misses. Assume the frequency of all loads and stores is 30%. How much faster a processor will run with a perfect cache that never missed.

**Answer:**

- Orignal:
  - Instruction fetches: 0.03 * 100 = 3 cycles(stall)
  - Data loads / stores: (stall) = 0.30 * 0.06 * 100 = 1.8 cycles(stall)

  Total miss penalty: 3 cycles + 1.8 cycles = 4.8 cycles

  CPI with memory stalls: 4 cycles + 4.8 cycles = 8.8 cycles

- Perfect:

  Only base CPI: 4 cycles

Speedup = Total CPI with original cache / Total CPI with perfect cache

8.8 / 4 = 2.2

2. (25 points) The execution of an instruction can be divided into five parts: instruction fetch (IF), register read (RR), ALU operation (EX), data access (MEM), and register write (RW). The following Table 1 shows the execution time of each part for several types of instructions, assuming that the multiplexors, and control unit have no delay.

| | Instruction fetch | Register read | ALU operation | Data access | Register write |
|---|---|---|---|---|---|
| Load word (lw) | 200 ps | 100 ps | 200 ps | 200 ps | 100 ps |
| Store word (sw) | 200 ps | 100 ps | 200 ps | 200 ps | |
| R-format (add, sub, and, or, slt) | 200 ps | 100 ps | 200 ps | | 100 ps |
| Branch (beq) | 200 ps | 100 ps | 200 ps | | |

If instructions are to be executed in a pipelined CPU with five pipeline stages, IF, RR, EX, MEM, RW where the pipeline stages execute the corresponding operations mentioned above.

(i) What is the cycle time of the pipelined CPU? What is the maximum working frequency?

(ii) What is the latency of executing the load-word instruction (lw) in the pipelined CPU?

(iii) What is the latency of executing the add instruction (add) in the pipelined CPU?

(iv) What is the maximum throughput of the pipelined CPU?

(i) The cycle time of the pipelined CPU is determined by the longest delay of the pipeline stages. In this case, it is 200 ps. The maximum working frequency is the reciprocal of the cycle time, which is 1/(200 ps) = 5 GHz.

(ii) The latency of executing the load-word instruction (lw) in the pipelined CPU is the time it takes for the instruction to go through all five pipeline stages. In this case, it is 200 ps + 100 ps + 200 ps + 200 ps + 100 ps = 800 ps.

(iii) The latency of executing the add instruction (add) in the pipelined CPU is also the time it takes for the instruction to go through all five pipeline stages. In this case, it is 200 ps + 100 ps + 200 ps + 0 ps + 100 ps = 600 ps.

(iv) The maximum throughput of the pipelined CPU is one instruction per cycle time, which is 1/(200 ps) = 5 billion instructions per second.

> A virtual memory system often implements a TLB to speed up the virtual-to-physical address translation. A TLB has the following characteristics. Assume each TLB entry has a valid bit, a dirty bit, a tag, and the page number. Determine the exact total number of bits to implement this TLB.
> - It is direct-mapped.
> - It has 16 entries.
> - The page size is 4 Kbytes.
> - The virtual address space is 4 Gbytes.
> - The physical memory is 1 Gbytes.

**Answer:**

- The TLB is direct-mapped and has 16 entries.
- The page size is 4 Kbytes = 2^12 bytes.
- The virtual address space is 4 Gbytes = 2^32 bytes.
- The physical memory is 1 Gbyte = 2^30 bytes.

1. Tag:
   1. The tag is used to determine which virtual page number is currently stored in a TLB entry. Given a virtual address space of 2^32 bytes and a page size of 2^12 bytes, we have a total of (2^32 / 2^12) = 2^20 pages.
   2. Since the TLB is direct-mapped and has 16 entries, each TLB entry can potentially map to one of (2^20 / 2^4) = 2^16 pages.
   3. the number of bits required for the tag is log2(2^16) = 16 bits per entry
2. Page Number:
   1. The page number refers to the physical page number, given a physical memory size of 2^30 and a page size of 2^12, we have (2^30 / 2^12) = 2^18 physical pages
   2. Therefore, the number of bits required for the PPN is log2(2^18) = 18 bits per entry
3. Valid and Dirty bit: Since each TLB entry has a valid bit and a dirty bit, we need 2 bits
4. We need (16 + 18 + 2) * 16(entries) = 576 bits to implement this TLB
   Total number of bits:
   32 bits (valid and dirty bits) + 256 bits (tag) + 288 bits (Page Number) = 576 bits

## Problems

1. What is F8EF - 1A0B when these values represent signed 16-bit hex numbers stored in sign-magnitude format?

```
F8EF = 1111 1000 1110 1111 -> negative => -30959_ten
1A0B = 0001 1010 0000 1011 => 6667_ten
F8EF - 1A0B = -30959 - 6667 = -37626
=> 37626 = 1001 0010 1111 1010
=> -37626 = 1 1001 0010 1111 1010 (overflow)
Ans: 1001 0010 1111 1010 = 92FA
```

2. Assume 168 and 78 are signed 8-bit decimal integers stored in two's complement format. Calculate 168+78 using saturating arithmetic.

```
168_ten = 1010 1000 =>(minus 1) 1010 0111 => (Reverse) 0101 1000 = 88 => -88
-88 + 78 = -10
```

1. Assume a color display using 8 bits for each of the primary colors (red, green, blue) per pixel, and a frame size of 4096x2160.

   (a) (5 points) What is the minimum size in bytes of the frame buffer to store a frame?
   8 bits = 1 byte
   4096*2160*3 = 8847360*3 = 26542080 bytes

   (b) (5 points) How long would it take, at a minimum, for the frame to be sent over a 100 Mbit/s network?
   26542080 * 8 / 100 * $2^{20}$ = 212336640/100*$2^{20}$ → 約 2.12 seconds

2. Assume a program requires the execution of 50x$10^6$ FP instructions, 110x$10^6$ INT instructions, 120x$10^6$ L/S instructions, and 16x$10^6$ branch instructions. The CPI for each type of instruction is 4, 1, 4, and 2, respectively. Assume that the processor has a 2 GHz clock rate.

   (a) (5 points) By how much must we improve the CPI of L/S instructions if we want the program to run two times faster?
   原本的執行時間 = 50*$10^6$*4/2*$10^9$ + 110*$10^6$/2*$10^9$ + 120*$10^6$*4/2*$10^9$ + 16*$10^6$*2/2*$10^9$ =411*$10^{-3}$ = 0.411 sec
   設 new L/S CPI = X
   0.411 / 2 = 50*$10^6$*4/2*$10^9$ + 110*$10^6$/2*$10^9$ + 120*$10^6$*X/2*$10^9$ + 16*$10^6$*2/2*$10^9$
   50*$10^6$*4/2*$10^9$ + 110*$10^6$/2*$10^9$ + 120*$10^6$*X/2*$10^9$ + 16*$10^6$*2/2*$10^9$ = 0.2055
   X = 0.575 → 將 CPI of L/S 設為 0.575
   Reduction of CPI = 4 – 0.575 = 3.425

   (b) (5 points) By how much is the execution time of the program improved if the CPI of INT and FP instructions is reduced by 40% and the CPI of L/S and Branch is reduced by 30%?
   The CPI for each type of instruction is 2.4, 1, 2.8, and 1.4
   New execution time = (25*2.5 + 55*1 + 60*2.8 + 8*1.4)*$10^{-3}$ = 296.7*$10^{-3}$ =0.2967 sec
   Reduction of execution time = 0.411 – 0.2967 = 0.1143 sec

3. (5 points) Provide the type and hexadecimal representation of the following instruction: lw $t1, 32($t2)
   I-type
   (10 0011) + (01010) + (01001) + (0000 0000 0010 0000) = 0X8d490020

4. (15 points) Translate the following C function to MIPS assembly code. Assume that n 0 is always hold when the function is called. Use a minimum number of instructions.

```
int fib(int n){
    if(n==0) return 0;
    elseif(n==1) return 1;
    elsereturn      fib(n      1)+fib(n      2);
}
```

```
fib :   addi $sp,$sp,-12
        sw $ra,8($sp)
        sw $s0,4($sp)
        sw $a0,0($sp)
        bne $a0,$0,L1
        add $v0,$0,$0
        j exit
L1:     addi $t0,$0,1
        bne $a0,$0,L2
        add $v0,$t0,$0
        j exit
L2:     addi $a0,$a0,-1
        jal fib
        add $s0,$v0,0
        addi $a0,$a0,-1
        jal fib
        add $v0,$v0,$s0
exit:   lw $a0,0($sp)
        lw $s0,4($sp)
        lw $s0,0($sp)
        addi $sp,$sp,12
        jr $ra
```

5. (5 points) Calculate 5EDB - 0CA4 when the values represent unsigned 16bit hexadecimal numbers. The result should be written in hexadecimal. Show your work.
   5EDB = 0101 1110 1101 1011
   0CA4 = 0000 1100 1010 0100
   5EDB – 0CA4 = 0101 0010 0011 0111

7. Write down the hexadecimal representation of a decimal number -31.75 in the following format. Show your work.

   (a) (5 points) IEEE 754 single precision format
   $31_{ten}$ = 1 1111$_{two}$
   $0.75_{ten}$ = 0.11$_{two}$
   $31.75_{ten}$ = 11111.11 = 1.111111 x $2^4$
   Sign = 1
   Exp = 4 + 127 = 131 = 10000011$_{two}$
   Mantissa = 1.11111100000000000000000 – 1 => 11111100000000000000000
   1100 0001 1111 1110 0000 0000 0000 0000 = 0XC1FE0000

   (b) (5 points) IEEE 754 double precision format
   $31_{ten}$ = 1 1111$_{two}$
   $0.75_{ten}$ = 0.11$_{two}$
   $31.75_{ten}$ = 11111.11 = 1.111111 x $2^4$
   Sign = 1
   Exp = 4 + 1023 = 1027 = 100 0000 0011$_{two}$
   Mantissa = 1.11111100000000000000000 – 1 => 1111 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000$_{two}$
   1 100 0000 0011 1111 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000= 0XC03FC00000000000

8. When processor designers consider a possible improvement to a processor datapath, a decision usually depends on the cost/performance trade-o . In Figure 2, the I-Mem, Add, Mux, ALU, Regs, D-Mem, and control blocks have latencies of 400 ps, 100 ps, 30 ps, 120 ps, 200 ps, 350 ps, and 100 ps, respectively, and costs of 1000, 30, 10, 100, 200, 2000, and 500, respectively. Consider the addition of a multiplier to the ALU. The addition will add 300 ps to the latency of the ALU and will add a cost of 600 to the ALU. The result will be 5% fewer instructions executed since we will no longer need to emulate MUL instructions.

   (a) (5 points) What is the clock cycle time with and without the improvement?

| Old | I-mem | Add | Mux | ALU | Regs | D-mem | Control |
|---|---|---|---|---|---|---|---|
| Latency | 400 | 100 | 30 | 120 | 200 | 350 | 100 |
| cost | 1000 | 30 | 10 | 100 | 200 | 2000 | 500 |

| New | I-mem | Add | Mux | ALU | Regs | D-mem | Control |
|---|---|---|---|---|---|---|---|
| Latency | 400 | 100 | 30 | 420 | 200 | 350 | 100 |
| cost | 1000 | 30 | 10 | 700 | 200 | 2000 | 500 |

   Pc → I-mem → Regs → Mux → ALU → D-mem → Mux → Regs
   Without improvement clock cycle time = 400+200+30+120+350+30+200 = 1330 ps
   With improvement clock cycle time = 1330+300 = 1630 ps

   (b) (5 points) What is the speedup achieved by adding the improvement?
   Improvement → 1630 * 0.95 = 1548.5
   1330 / 1548.5 = 0.86
   New is 14% slower than old one

9. Table 2 shows the latency of each stage of the datapath in a processor. Table 3 shows the breakdown of instructions in a program executed by the processor.

Table 2: Latency of the datapath (ps)

| Stage | IF | ID | EX | MEM | WB |
|---|---|---|---|---|---|
| Latency | 250 | 350 | 150 | 300 | 250 |

Table 3: Breakdown of a program (%)

| Instruction | alu | beq | lw | sw |
|---|---|---|---|---|
| Ratio | 40 | 20 | 25 | 15 |

   (a) (10 points) What is the clock cycle time in a pipelined and non-pipelined processor?
   Pipelined clock cycle time = 350 ps
   Non-pipelined clock cycle time = 250+350+150+300+250 = 1300 ps

   (b) (10 points) What is the total latency of an LW instruction in a pipelined and non pipelined processor?

| Instruction | IF | ID | EX | MEM | WB | Total |
|---|---|---|---|---|---|---|
| alu | 250 | 350 | 150 | | 250 | 1000 |
| beq | 250 | 350 | 150 | | | 750 |
| lw | 250 | 350 | 150 | 300 | 250 | 1300 |
| sw | 250 | 350 | 150 | 300 | | 1050 |

   Total latency of LW instruction
   non-pipeline： 1300 (ps)
   pipeline：350*5 = 1750 (ps)

   (c) (5 points) Assuming there are no stalls or hazards, what is the utilization of the write-register port of the "Register" unit?

   no stalls or hazards => the utilization of the write-register = ALU + LW = 40% + 25% = 65%

6. (20 points) Calculate the product of hexadecimal unsigned 8bit integers 13 and B using the hardware shown in Figure 1. The size of the Multiplicand, ALU, and Product is modified to 6, 6, and 12 bits, respectively. Complete Table 1 to show the contents of each register in each step.

Table 1: Multiplication operations

| Step | Action | Multiplicand | Product/Multiplier |
|---|---|---|---|
| 0 | Initial values | 010 011 (13) | 000 000 001 011 (B) |
| 1 | Prod = Prod + Mcand<br>Right shift Prod | 010 011<br>010 011 | 010 011 001 011<br>001 001 100 101 |
| 2 | Prob = Prob+Mcand<br>Right shift Prob | 010 011<br>010 011 | 011 100 100 101<br>001 110 010 010 |
| 3 | Lsb = 0,no operation<br>Right Shift Prob | 010 011<br>010 011 | 001 110 010 010<br>000 111 001 001 |
| 4 | Prob = Prob+Mcand<br>Right shift Prob | 010 011<br>010 011 | 011 010 001 001<br>001 101 000 100 |
| 5 | Lsb = 0,no operation<br>Right Shift Prob | 010 011<br>010 011 | 001 101 000 100<br>000 110 100 010 |
| 6 | Lsb = 0,no operation<br>Right Shift Prob | 010 011<br>010 011 | 000 110 100 010<br>000 011 010 001 |

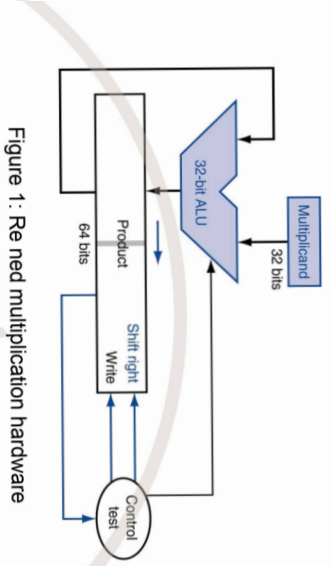



Figure 1: Re ned multiplication hardware

Consider a virtual memory system with the following characteristics:
- Page size of 64 KB= 2^16 bytes.
- Page table and page directory entries of 8 bytes per entry.
- Inverted page table entries of 16 bytes per entry.
- 31 bit physical address, byte addressed.
- Two-level page table structure (containing a page directory and one layer of page tables).

What is the size (in number of address bits) of the virtual address space supported by the above virtual memory configuration?

**Answer:**

- Offset: 16 bis
- Page Table: 64KB / 8 = 2^13 page table entries, 13 bits
- Page Directory: 64KB / 8 = 2^13 page directory entries, 13 bits

Total virtual address bits = Offset bits + Page Table bits + Page Directory bits

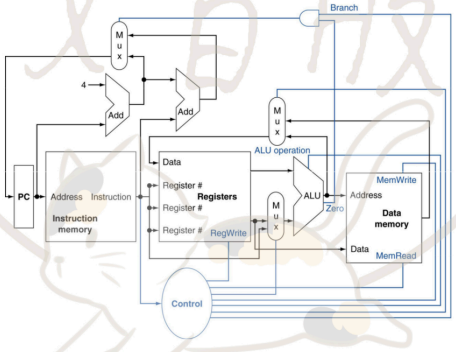
16 + 13 + 13 = 42 bits

What is the maximum required size (in KB, MB, or GB) of an inverted page table for the above virtual memory configuration?

**Answer:**

Number of frames = Physical memory size / Page size
= 2^31 bytes / 2^16 bytes
= 2^15

Inverted page table size = Number of frames * Size per entry
= 2^15 * 16 bytes
= 2^19 bytes
= 512 KB




(c) (5 points) What is the cost with and without the improvement?

Without :
Cost = 1000 + 2*30 + 3*10 + 100 + 200 + 2000 + 500 = 3890

With :

Cost = 3890 + 600 = 4490

Opc

| Instruction | Opcode/Function | Syntax |
|---|---|---|
| add | 100000 | ArithLog |
| addu | 100001 | ArithLog |
| addi | 001000 | ArithLogI |
| addiu | 001001 | ArithLogI |
| and | 100100 | ArithLog |
| andi | 001100 | ArithLogI |
| div | 011010 | DivMult |
| divu | 011011 | DivMult |
| mult | 011000 | DivMult |
| multu | 011001 | DivMult |
| nor | 100111 | ArithLog |
| or | 100101 | ArithLog |
| ori | 001101 | ArithLogI |
| sll | 000000 | Shift |
| sllv | 000100 | ShiftV |
| sra | 000011 | Shift |
| srav | 000111 | ShiftV |
| srl | 000010 | Shift |
| srlv | 000110 | ShiftV |
| sub | 100010 | ArithLog |
| subu | 100011 | ArithLog |
| xor | 100110 | ArithLog |
| xori | 001110 | ArithLogI |
| lhi | 011001 | LoadI |
| llo | 011000 | LoadI |

Table

| Instruction | Opcode/Function | Syntax |
|---|---|---|
| slt | 101010 | ArithLog |
| sltu | 101001 | ArithLog |
| slti | 001010 | ArithLogI |
| sltiu | 001001 | ArithLogI |
| beq | 000100 | Branch |
| bgtz | 000111 | BranchZ |
| blez | 000110 | BranchZ |
| bne | 000101 | Branch |
| j | 000010 | Jump |
| jal | 000011 | Jump |
| jalr | 001001 | JumpR |
| jr | 001000 | JumpR |
| lb | 100000 | LoadStore |
| lbu | 100100 | LoadStore |
| lh | 100001 | LoadStore |
| lhu | 100101 | LoadStore |
| lw | 100011 | LoadStore |
| sb | 101000 | LoadStore |
| sh | 101001 | LoadStore |
| sw | 101011 | LoadStore |
| mfhi | 010000 | MoveFrom |
| mflo | 010010 | MoveFrom |
| mthi | 010001 | MoveTo |
| mtlo | 010011 | MoveTo |
| trap | 011010 | Trap |