University of Newcastle
School of Electrical Engineering and Computer Science

COMP2240 - Operating Systems
Workshop 1 Solution
Topics: Hardware Review & Operating System Overview

1. Consider a hypothetical 32-bit microprocessor having 32-bit instructions composed of two fields (Figure 1). The first byte contains the opcode and the remainder an immediate operand or an operand address.

a) What is the maximum directly addressable memory capacity (in bytes)?
b) Discuss the impact on the system speed if the microprocessor bus has
   I. 32-bit local address bus and a 16-bit local data bus, or
   II. 16-bit local address bus and a 16-bit local data bus.
c) How many bits are needed for the program counter and the instruction register?

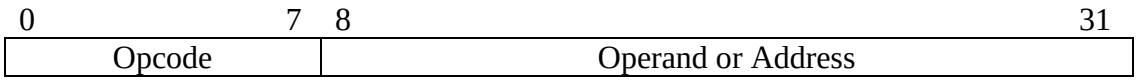


Figure 1: Instruction Format of the 32-bit Hypothetical Machine

Answer:

a) 2^{24} = 16 Mbytes

b) Referred to Figure 1.

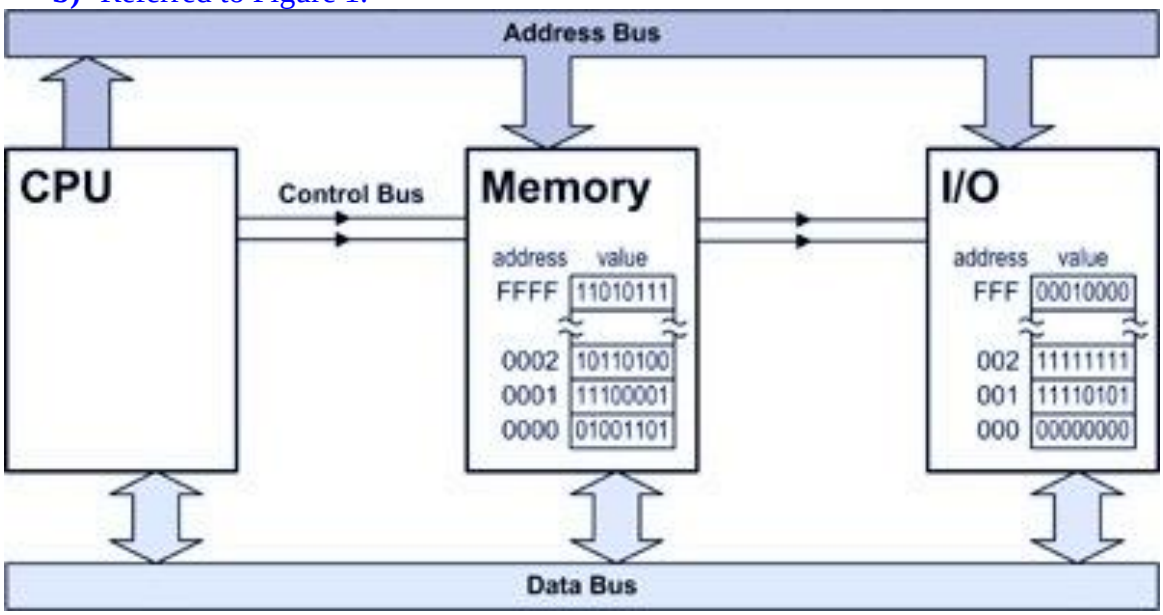


Figure 1: Typical Microprocessor Bus Structure (Source: http://www.talktoanit.com/A+/aplus-website/lessons-io-principles.html)

I. If the local address bus is 32 bits, the whole address can be transferred at once and decoded in memory. However, since the data bus is only 16 bits, it will require 2 cycles to fetch a 32-bit instruction or operand.
II. The 16 bits of the address placed on the address bus can't access the whole memory. Thus

a more complex memory interface control is needed to latch the first part of the address and then the second part (since the microprocessor will send it in two steps). For a 32-bit address, one may assume the first half will decode to access a "row" in memory, while the second half is sent later to access a "column" in memory. In addition to the two-step address operation, the microprocessor will need 2 cycles to fetch the 32 bit instruction/operand.

c) The program counter must be at least 24 bits. Typically, a 32-bit microprocessor will have a 32-bit external address bus and a 32-bit program counter, unless on-chip segment registers are used that may work with a smaller program counter. If the instruction register is to contain the whole instruction, it will have to be 32-bits long; if it will contain only the op code (called the op code register) then it will have to be 8 bits long.

2. A DMA module is transferring characters to main memory from an external device transmitting at 9600 bits per second (bps). The processor can fetch instructions at the rate of 1 million instructions per second.
By how much will the processor be slowed down due to the DMA activity? (You can ignore the data read/write operations and assume that the processor only fetches instructions.)

Answer:
Let us ignore data read/write operations and assume the processor only fetches instructions. Then the processor needs access to main memory once every microsecond. The DMA module is transferring characters at a rate of 1200 characters per second (by assuming 8 bit per char or byte), or one every 833 μs.

The DMA therefore "steals" every 833rd cycle. This slows down the processor approximately $\frac{1}{833} \times 100\% = 0.12\%$.

Alternate Solution:
Let us ignore data read/write operations and assume the processor only fetches instructions. Then the processor needs access to main memory once every microsecond.

9600 bps = 9600/8 = 1200 char per sec (by assuming 8 bit per char or byte).
Slow down $= \frac{1200}{1\times10^6} \times 100\% = 0.12\%$

3. Suppose that a large file is being accessed by a computer memory system comprising of a cache and a main memory. The cache access time is 60ns. Time to access main memory (including cache access) is 300ns. The file can be opened either in read or in write mode. A write operation involves accessing both main memory and the cache (write-through cache). A read operation access either only the cache or both the cache and main memory depending upon whether the access word is found in the cache or not. It is estimated that read operations comprise of 80% of all operations. If the cache hit ratio for read operation is 0.9, what is the average access time of this system?

Answer:
Cache access time, t_c = 60 ns
Main memory access time, t_m = 300 ns
Hit ratio, h = 0.9
Percentage of read operations, p_r = 80% = 0.8

Percentage of write operations, $p_w = 100 - 80 = 20\% = 0.2$

Average access time of read operations = $ht_c + (1 - h)t_m$
$= 0.9 \times 60 + (1 - 0.9) \times 300$
$= 84$ ns
Effective time, t_{eff} for read operations = 0.8×84
$= 67.2$ ns

Average access time of write operations = 300 ns
Effective time, t_{eff} for write operations = 0.2×300
$= 60$ ns

Total time = t_{eff} for read + t_{eff} for write = $67.2 + 60$
$= 127.2$ ns

4. How are iOS and Android similar? How are they different?
Answer:
Similarities

- Both are based on existing kernels (Linux and Mac OS X).
- Both have architecture that uses software stacks.
- Both provide frameworks for developers.

Differences

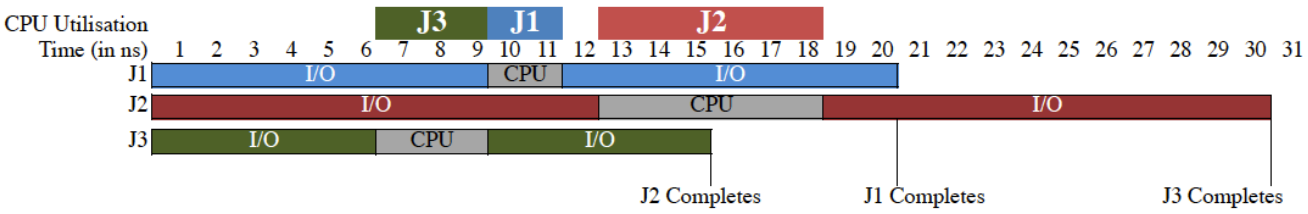
- iOS is closed-source, and Android is open-source.
- iOS applications are developed in Objective-C, Android in Java.
- Android uses a virtual machine, and iOS executes code natively.

5. In a batch operating system, three jobs J1,J2, and J3 are submitted for execution. Each job involves and I/O activity, a CPU time, and another I/O activity.
Job J1 requires a total of 20 ms, with 2 ms CPU time.
J2 requires 30 ms total time with 6 ms CPU time.
J3 requires 15 ms total time with 3 ms CPU time.
What will be the CPU utilization for **uniprogramming** and **multiprogramming**?

Answer:
Uniprogramming: the three jobs are executed sequentially such that a job is executed only when the previous job runs into completion.
Hence, the total time required for completion of the jobs = 20 + 30 +15 = 65 ms
Total execution time in CPU = 2 + 6 + 3 = 11 ms
CPU utilization = 11/65 = 0.1692= 16.92%

Multiprogramming: when one job is waiting for I/O, another job may execute its I/O.
I/O time for first job, J1 = (Total time – processor time)/2 = (20 –2)/2 = 9 ms
I/O time for second job, J2 = (Total time – processor time)/2 = (30 –6)/2 = 12 ms
I/O time for third job, J3 = (Total time – processor time)/2 = (15 –3)/2 = 6 ms

The execution in a multiprogramming environment if all the jobs are submitted at the same time is as follows:



From the above diagram, it is evident that total time = 30 ms
Total execution time in CPU = 11 ms
CPU utilization = 11/30 = 36.67%

Supplementary problems:

S1. Direct memory access (DMA) is used for high-speed I/O devices in order to avoid increasing the CPU's execution load.

a) How does the CPU interface with the device to coordinate the transfer?
b) How does the CPU know when the memory operations are complete?
c) The CPU is allowed to execute other programs while the DMA controller is transferring data. Does this process interfere with the execution of the user programs? If so, describe what forms of interference are caused.

Answer:

a) To initiate a DMA transfer, the CPU first sets up the DMA registers, which contain a pointer to the source of a transfer, a pointer to the destination of the transfer, and a counter of the number of bytes to be transferred. Then the DMA controller proceeds to place addresses on the bus to perform transfers, while the CPU is available to accomplish other work.
b) Once the entire transfer is finished, the DMA controller interrupts the CPU.
c) Both the CPU and the DMA controller are bus masters. A problem would be created if both the CPU and the DMA controller want to access the memory at the same time. Accordingly, the CPU should be momentarily prevented from accessing main memory when the DMA controller seizes the memory bus. However, if the CPU is still allowed to access data in its primary and secondary caches, a coherency issue may be created if both the CPU and the DMA controller update the same memory locations.

S2. Suppose a stack is to be used by the processor to manage procedure calls and returns. Can the program counter be eliminated by using the top of the stack as a program counter?

Answer:
Yes, if the stack is only used to hold the return address. If the stack is also used to pass parameters, then the scheme will work only if it is the control unit that removes parameters, rather than machine instructions. In the latter case, the processor would need both a parameter and the PC on top of the stack at the same time.

S3. Consider a hypothetical microprocessor generating a 16-bit address (e.g., assume that the program counter and the address registers are 16 bits wide) and having a 16-bit data bus.

a) What is the maximum memory address space that the processor can access directly if it is connected to a "16-bit memory"?
b) What is the maximum memory address space that the processor can access directly if it

is connected to an “8-bit memory”?

c) What architectural features will allow this microprocessor to access a separate “I/O space”?

d) If an input and an output instruction can specify an 8-bit I/O port number, how many 8-bit I/O ports can the microprocessor support? How many 16-bit I/O ports?

Explain.

Answer:

In cases **(a)** and **(b)**, the microprocessor will be able to access 2^{16} = 64K bytes; the only difference is that with an 8-bit memory each access will transfer a byte, while with a 16-bit memory an access may transfer a byte or a 16-byte word.

For case **(c)**, separate input and output instructions are needed, whose execution will generate separate "I/O signals" (different from the "memory signals" generated with the execution of memory-type instructions); at a minimum, one additional output pin will be required to carry this new signal.

For case **(d)**, it can support 2^8 = 256 input and 2^8 = 256 output byte ports and the same number of input and output 16-bit ports; in either case, the distinction between an input and an output port is defined by the different signal that the executed input or output instruction generated.

S4. Suppose that we have a multiprogrammed computer in which each job has identical characteristics. In one computation period, T, for a job, half the time is spent in I/O and the other half in processor activity. Each job runs for a total of N periods. Assume that a simple round-robin scheduling is used, and that I/O operations can overlap with processor operation. Define the following quantities:

- Turnaround time = actual time to complete a job
- Throughput = average number of jobs completed per time period T
- Processor utilization = percentage of time that the processor is active (not waiting)

Compute these quantities for one, two, and four simultaneous jobs, assuming that the period T is distributed in each of the following ways:

a) I/O first half, processor second half

I/O first and fourth quarters, processor second and third quarter

Answer:

In case of 1 job: CPU utilization is 50% for both (a) and (b).
In case of two jobs: it is N/(N+0.5) for (a) and N/(N+0.25) for (b).
In case of four jobs: it is 2N/(2N+0.5) for (a) and 2N/(2N+0.25) for (b).

Assume that all jobs arrived at the same time, so we count the turnaround time of all jobs from the same moment.

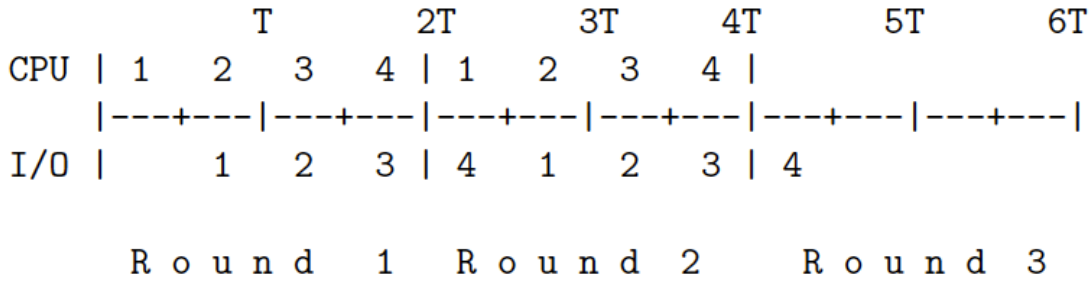
1 job :
The turnaround time in case of 1 job is N*T.

2 jobs :
In case (a), the individual turnaround times are NT, (N+0.5)T and the average is (N+0.25)T. In case (b), the average is 0.5(NT + (N+0.25)T) = T(N+1/8).

4 jobs:

In case (a), the turnaround times for the individual jobs are 2NT-T, (2N-0.5)T, 2NT, (2N+0.5)T and the average is (2N-0.25)T. In case (b), the individual turnaround times are (2N-0.5)T, (2N-0.25)T, 2NT, (2N+0.25)T and the average is (2N+7/8)T.

All these figures are obtained by drawing and then examining the execution diagrams. For instance, in case of 4 jobs and the execution pattern (a), such a diagram might looks as follows:



We see that each round of execution where all jobs finish their time period T takes 2T+0.5T. The next round will finish at 4T+0.5T, the following at 6T+0.5T, etc. There- fore, executing all jobs will take 2TN+0.5T time units. Job 4 ends at this time, job 3 ends 0.5T earlier, job 2 ends another 0.5T before that, etc.