**PREM KRISHNA CHETTRI**

**Computer Architecture Assignment 1 Submission Date: 21st Sept ‘15**

**Solution 1** :

Dalvik uses optimization on the .class file that Java compiler has generated by compiling .Java file and creates a .dex file. So, if we look at the virtual machine of Dalvik, we find that the instruction set that Dalvik VM is suppose to executed is much more optimized and is designed specially for the systems which has a memory and processors limitation. However, this comes with the cost of additional steps to re-compile with and additional piece of dx compiler software.

In terms of vm architecture, Dalvik uses the register based architecture and is designed to facilitate the executing of independent processes for each application, which makes its instruction set complex. However number of instructions has been reduced significantly as everything lies within the registers and don’t have to manage the different load and decode instructions like stack based architecture. In terms of instruction set, Dalvik uses the fixed 16 bit instructions compared to 8 bit instructions for standard Java bytecode. The advantage of this is, as local variable commonly picked up by four bit virtual registers, this optimization gives a significant advantage to the Dalvik’s instruction count and interpretational speed. Dalvik, design is highly coupled with hardware, considering these hardware specifics, where it’s instructions are going to get executed, each version of Dalvik can significantly take an advantage of its hardware and its availability. So the approach of the Dalvik gives certain advantage to the devices and can run multiple vm instances for each application and process efficiently.

Java was designed in different timeline and with different purpose. It is the generic approach to resolve how to make machine code independent of hardware with the help of generation of bytecode. Architecturally, it uses a stack-based architecture, which means we will have a significantly larger instruction set to execute the task and so it has a different instruction for memory and data (load and manipulate the data and all). It is designed for meeting the purpose for general systems not as specific as Dalvik and so it bytecode is not as optimized (which dalvik took care later). It is also not highly coupled as Dalvik as it’s not designed specifically for any particular hardware.

Java virtual machine works directly on the .class file that has been generated by java compiler. So it does not have the further optimization step. All instruction are basically being stacked within the system internals stack and for each instruction, it has to fetch the value / decode it and then go for the execution. Thereby increasing the number of instructions as well as hampering execution speed.

**Solution 2 :**

Spatial Locality:

It is observed than one of the intrinsic behaviors of most computer programs is that the data elements that gets executed lies within the predictable offset of the currently executing memory location. In-order to reduce the overhead of calling the slower memory module every time, CPU architecture is generally designed to take an advantage of this property and predict the future location and cache those data into much faster CPU cache location ahead of its execution. Such a kind of caching is called the spatial locality. So, now the CPU does not have to wait longer, for executing the data element if it exists in its local cache, and it will significantly effect is various factors regarding performance of the CPU.

Temporal Locality:

Computer programs often tend to reuse the data, which it recently has executed. The interdependency of programs occurs as we build bigger programs and functionality. Generally instructions, functions and programs take much bigger time to fully execute in one single cycle, and so it has to process that earlier executed data to compute the next instruction. In doing so, if we always have to read / write to the slower main memory every time, then we will be compromising the performance of the CPU. So, the CPU is designed to cache data, which are processed earlier and reuse it in later executions. Such a type of caching is called temporal locality and it helps by increasing the access time of data element from the cache if a data element exist within its cache.

**Solution 3**

Work = power \* time. So we will get the total work done by the Power (P) in the time (t) and is measured in joules.

**Solution 4**

Advantages

1. Resource Utilization : Considering the instruction set, which can be pipelined, the amount of resource utilization is significantly higher compared to non-pipelined architecture.
2. Power Advantage : With the increase in resource utilization, we are reducing the static power consumption(which is at-least 25% of the total power).
3. Throughput advantage : Pipelining has a major advantage in throughput of the entire workload.
4. Time to execute task is significantly reduced.

**Solution 5**

Speed up is a ratio between the time required without pipeline and the time with the pipeline.

So Speedup = ( N \* K \* T(k)) / ( K \* T(k) + (N-1) \* T (k)) ,

Where K is the number of pipleline stages,

N is number of instructions

T (k) is the timestamp at stage k.

When N is given a value close to infinity.

Speedup = approaches to K, when N approaches infinity.

**Solution 5**

Pipeline registers are memory modules that are inserted in between the pipeline stages and are clocked synchronously along with the different stages of the pipelines. The main purpose of these registers is to hold the data till all the stages have processed their portion of the data and is ready to process for the next stage.

Architecture registers, acts as the temporary fast access memory module, which helps in speeding up the memory access time and performance of the CPU.

**Solution 6.1**

Path 1 uses 2 clock cycles where as Path 2 uses 4 clock cycles. Hence, Path 1 uses minimum clock period.

**Solution 6.2**

As, each stage of pipeline may take different clock signal to process the data. In order to allow each stage to completely process, we have to make sure that we provide sufficient amount of time for each stage. When we use shorter than the maximum signal propagation time through the circuit, there comes a situation where we may lose data, as the data hasn’t been properly process by all the processing units due to the clock skew.

**Solution 6.3**

Electrical signals are basically a movement of electrons in a conducting circuit, which will have its physical limitation.

**Solution 7**

Pipeline stages sometimes goes unused and that is what we usually called as pipeline stall. This occurs because of the delay in the executing of earlier stages in the pipeline processing. As it happens, when a stage does not process any data due to the interdependency from earlier stages, we lose the resource utilization and have significant effect on the performance of the system.

**Solution 8:**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Pipeline logic delay (FO4)*** | ***Total pipeline stage delay (FO4)*** | ***Clock rate (1/FO4)*** | ***Maximum speedup vs. 16 FO4*** |
| 16 | 18 | 1 / 18 | 1.0 |
| 8 | 10 | 1 / 10 | 1.8 |
| 4 | 6 | 1 / 6 | 3 |
| 2 | 4 | 1 / 4 | 4.5 |
| 1 | 3 | 1 / 3 | 6 |

**Solution 9**

1. What does it mean if there are two X’s in a row?

Solution : It means that a particular resource is being utilized 2 times for an instruction execution at two different time stamp.

1. What does it mean if there are two X’s in a column?

Solution : It means that two different resources are being utilized at a particular instance of time during the exe

**Solution 10** :- To achieve this, we need to use two parallel registers accepting each one of its input as the one of the 2 inputs operand for each 2 register. Each of this input will acts an independent input value of our potential 2 input operands. When we intend to do some processing, we can use an additional circuit to determine if it is a 2-operand input, on which case, we can pass the signal through a parallel set of registers or else we can use only 1 register file. Now if logical computation uses the 2-operand inputs, we can use a multiplexer them to combine the values of the two registers and feed through the next pipeline stage.

As far as write back operation is concern, we can use the de-multiplexer and feed the write back output back to the respective registers in the given architecture as for each write back.

The implication of this is the complex architectural design. Also, it involves the additional component like multiplexer, de-multiplexer and a decision board. With addition of these components, it will drive the higher cost of the total architecture as well as the power consumption. It will always add up to the performance of the system as for each of the input value, we have to pass the signal through various additional circuits, which take its own time to evaluate the signal processing.