ACA Assignment 2

Reading and Research Assignment

The author in this paper presents the comparison between an example implementation of the RISC and CISC architecture on 9 among 10 benchmarks released SPEC by considering the architectural similarities among 2 different machines (MIPS M/2000 and Digital VAX 8700) and presenting the advantage of RISC having over CISC. RISC factor(ratio of instruction set to CPI) for the above architectures was calculated. The author has assumed the compilers of both the machines is the same and the cycle time is not dependent on the architecture. The main criteria they have considered for the comparison is the product of number of instructions executed, the mean number of cycles needed, the cycle time.

The cycle time may be affected by the system hardware, instruction set architecture. The CPI also depends on the instruction set architecture, like VAX where the individual instructions whose execution would be requiring 100 of cycles in parallel whereas the RISC would accomplish the same by in 1 or 2 cycles.

Even though both the similarities in their cycle times there are few advantages which are exhibited by the MIPS. The MIPS allows the advantage of overlap of floating point instructions over VAX, because MIPS architecture loads and stores instructions whereas the VAX uses the operands

The results of the comparison made in the paper were, the instruction set ratio of the MIPS to the VAX was always greater than 1 and less than 4 with geometric mean of 2.77, the CPI ratio was between 3 and 10.45 with geometric mean of 5.77, whereas the RISC factor ranged from 2 to 4 with geometric mean of 2.66

R2b

The author of this paper demonstrates how much ILP is in use and how various hardware and software architectures are employed to maximize its use. ILP is the count of how many instructions are running in parallel at an instance during the compile time. The major challenge in attaining ILP is dependencies in the program. There are diverse types of dependencies effect the parallelism such as true data, anti dependency, outpt dependenci, and contrl dependemcy.

To achieve the various level of parallelism various techniques have been incorporated such as Register Renaming, branch prediction, alias analysis, and simultaneous speculative execution. In order to rank these different techniques, we need to them consider under 7 models. Register renaming is a hardware technique to achieve ILP here the hardware dynamically assigns a new name to every register which is used to store the information as to achieve parallelism. As the quality of the model increases from stupid to fair, there is no effect on the parallelism achieved, while when the model varies between good and superb the effect increases exponentially, when the model is a perfect the parallelism achieved is also near to perfect.

Alias analysis occurs when two memory address access a common location. This technique is done because the references outer to heap can be solved by compiler. To reduce the dependencies the alias analysis is done by the compiler. The different types of alias analysis which our system does is perfect alias, no alias, alias by instruction inspection, alias analysis by compiler. Now considering the ranking for the 7 abstract models, there is no effect on the stupid model whereas it increases to inspect to the poor model and is perfect for the remaining models.

Branch prediction is used in computer architecture to improve the ILP by routeing the instructions and sending them across to various branches so that multiple instructions can be executed parallelly, the main purpose of the branch prediction is to effectively use the pipeline. The branches are predicted by plotting on a graph called branch profile. The effect of branch prediction on the 7 models is, for the stupid model there is no effect but as the quality of the model increases the parallelism increases exponentially.

Although these techniques are employed to achieve a higher ILP, they have been some drawbacks due to these we should be aware in advance as to how much these features help in improving the ILP. The latency also doesn’t have a direct impact on the parallelism, by improving the latency sometimes the ILP increases and sometimes it decreases. Theoretically, even the best model has low response to parallelism, which clearly depicts the fact that the parallelism of the programs is much more dependent on the other factors than just the branch predictability. If we contradict with the assumption that the registers are not infinite then the effect of the registers on the ILP is also limited.

2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | SATA | SCSI | PCI-X | AGP |
| DATA WIDTH | 16 bits | 16 bits | 64 bits | 32 bits |
| CLOCK RATE | 3Ghz | 5MHz | 533 MHz | 66 MHz |
| BANDWIDTH | 300MBps | 40MBps | 1GBps | 254 MBps |

----------------------------------------------------------------------------------------------------------------

Exercise

1) MTTF- (Mean time to failure)

MTTF = 35

Number of computers = 10,000

Let the MTTF for all computers = “x”

MTTF = (Total hours of all computers worked) / (Number of computers failed)

Total time of all computes worked = 10,000\* x

Number of computers failed = ¼ \* (10,000) = 2500

MTTF =

35 = 10,000 \* x / (2500)

x = 8.75 days = 9 days(approx.)

----------------------------------------------------------------------------------------------------------------

2) SPEC ratio of Opteron = 20.86

SPEC ratio of Iteanium 2 = 27.12

SPEC ratio of Opteron / SPEC ratio of Iteanium 2 =

(Performance of Opteron) / (Performance of Iteanium 2)

20.86 / 27.12 = (Performance of Opteron) / (Performance of Iteanium 2)

0.769 = (Performance of Opteron) / (Performance of Iteanium 2)

Therefore, the performance of Iteanium 2 is better than Performance of Opteron

The weighted average of execution time ratios for Opteron to Iteanium 2 is =

(0.5\*(73.6/86.9)) +(0.2\*(94/50.9)) +(0.3\*(123/68.8))

= 1.320

----------------------------------------------------------------------------------------------------------------

3) Given

Misprediction penalty = 4 cycles

Buffer miss Penalty = 3 cycles

Branch Penalty = 2 cycles

Hit rate = 90% = 0.9

Accuracy rate = 80% = 0.8

Branch Frequency = 10% = 0.1

Base CPI without branch stalls = 1

CPI for Fixed two cycle branch penalty = base CPI+ (branch frequency \* branch penalty)

= 1 + (0.1\*2)

= 1 + 0.2

= 1.2

CPI for Deeply Pipelined Processor = base CPI + (branch frequency\*(miss in buffer + misprediction)

Miss in buffer = (0.1) \*3=0.3

Misprediction = 0.9 \*0.2\*4= 0.72

* CPI for Deeply Pipelined Processor = 1+ 0.1(0.3+0.72) =1.102

Speedup = (CPI for fixed 2 cycle branch penalty) / (CPI for deeply Pipelined Processor)

= 1.2 / 1.102 = 1.088

The performance of the branch buffer is better than the 2-fixed cycle by 1.088

----------------------------------------------------------------------------------------------------------------

4) Given

Type of architecture = 5 stage single pipeline

The number of cycles for LW and SW are 1+2

The number of cycles for branches are 1+1 cycles

In a pipelining model the instruction the instructions are not executed until all other operands are not completed.

The Pipeling has the 5 stages i.e Instruction Fetch(F), Instruction Decode (D), Instruction Execute (X), Memory Access(M), Write Back(W)

F- Instruction Fetch

D- Instruction Decode

X- Instruction Execute

M- Memory Access

W- Write Back

Here the instruction goes through the first 3 levels and the Memory access takes 1+2 cycles due to latency

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Loop | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| LW R3,0(R0) | F | D | X | M | - | - | W |  |  |  |  |  |  |  |  |  |  |  |  |
| LW R1,0(R3) |  | F | D | - | - | - | X | M | - | - | W |  |  |  |  |  |  |  |  |
| ADD1 R1, R1#1 |  |  | F | - | - | - | D | - | - | - | X | M | W |  |  |  |  |  |  |
| SUB R4,R3,R2 |  |  |  |  |  |  | F | - | - | - | D | X | M | W |  |  |  |  |  |
| SW R1,0(R3) |  |  |  |  |  |  |  |  |  |  | F | D | X | M | - | - | W |  |  |
| BNZ R4,LOOP |  |  |  |  |  |  |  |  |  |  |  | F | D | X | - | - | M | W |  |
| LW R3,0(R0) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D |  |

a) Due to the overhead there has been a loss of 4 cycles. Without bypassing, the output of the SUB instruction would not been possible until the W phase. Therefore the additional 4 clock cycles at the end of the loop. The first instruction could not be started till the complete loop has been completed.

b) With the static predictor 2 cycles have been lost. The static predictor has the capability of estimating backward branch at the decode stage. This requires the F and D to find out. Therefore 2 cycles are lost.

c)With the use of the dynamic predictor 2 cycles are lost. The dynamic predictor stores the previous branch instructions which were calculated earlier. So the prediction occurs in the same cycle

----------------------------------------------------------------------------------------------------------------

5 a) Not Enough Data since the count of cycles of other instructions set and CPI is not given

5 b) Not enough data since the cycle count of other instruction set and CPI is not given

5 c)

Given data:

We have assumed same instruction count for both the machines

Speed up ratio is given = 1.1

Speed up ratio = (Execution time of machine1) / (Execution time of machine 3)

1.1 = (IC\*CPI of machine 1 / clock rate of machine 1) / (IC \* CPI of machine3 / clock rate)

1.1 = (IC \*800 / 1000) / (IC \* CPI of machine 3 / 900)

Since the instruction count is same for both the machines

CPI of machine 3 = 654 cycles

----------------------------------------------------------------------------------------------------------------

CASE STUDY

6)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| LW R2,0(R1) | IF | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |  |  |
| LABEL1: BEQ R2, R0, LABEL2 |  | IF | ID | ID | EX | MEM | WB |  |  |  |  |  |  |  |  |
| LABEL2: SW R1, 0(R2) |  |  | IF | IF | ID | - | - | - |  |  |  |  |  |  |  |
| INSTRUCTION AFTER SW |  |  |  |  | IF | - | - | - | - |  |  |  |  |  |  |
| LW R3, 0(R2) |  |  |  |  |  | IF | ID | EX | MEM | WB |  |  |  |  |  |
| BEQ R3, R0, LABEL1 |  |  |  |  |  |  | IF | ID | ID | EX | MEM | WB |  |  |  |
| LABEL1: BEQ R2, R0, LABEL2 |  |  |  |  |  |  |  |  | IF | IF | ID | EX | MEM | WB |  |
| LABEL2: SW R1, 0(R2) |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | MEM | WB |