PROGRAMMING ASSIGNMENT REPORT

1)

Benchmark	Total no. of instructions	Load (%)	Store (%)	Unconditional Branch (%)	Conditional Branch (%)	Integer Computation (%)	Floating Point Computation (%)
anagram	25597555	25.36	9.93	4.46	10.30	44.63	5.31
test-math	46490	17.17	10.47	3.83	10.80	55.61	2.00
bzip	4000000000	23.10	11.72	1.34	11.44	52.41	0.00
gcc	4000000000	48.79	16.91	1.36	5.99	26.94	0.01
mcf	4000000000	37.67	7.67	5.10	16.16	33.39	0.02

2) The configuration of machine A is as follows (from the config_a.cfg file) is as follows:

- Total number of integer ALU's = 1
- Total number of integer multipliers/dividers = 1
- Total number of integer Memory system ports to CPU = 1
- Total number of floating point ALU's = 1
- Total number of floating point multipliers/dividers = 1
- Memory access bus width = 8 bytes
- Load/store queue size = 8
- Instruction Fetch queue size = 1
- Instruction decode width = 1 instruction/cycle.
- Instruction issue width = 1 instruction/cycle.
- Register update unit size= 1
- Machine follows in order execution.
- Bimod type of branch predictor

IPC of the mcf program on machine A is 0.4879 (from sim_config_a.out file).

```
@ 2299
aw = `0x12000b6ac: lda r3,-4080(r3)'

 [IF]      [DA]      [EX]      [WB]      [CT]
  aw        av

@ 2300
ax = `0x12000b6b0: s8addq r10,r3,r3'

 [IF]      [DA]      [EX]      [WB]      [CT]
  ax        aw        av

@ 2301
ay = `0x12000b6b4: ldq r3,0(r3)'

 [IF]      [DA]      [EX]      [WB]      [CT]
  ay        ax        aw        av

@ 2302
az = `0x12000b6b4: [internal ld/st]'
ba = `0x12000b6b8: lda r3,8(r3)'

 [IF]      [DA]      [EX]      [WB]      [CT]
  ba        ay        ax        aw        av
            az
```

Yes, data forwarding is implemented. The above picture, shows two instructions “aw” and “ax” where there is a chance for RAW hazard to take place. But if we look at the clock cycle 2302, we can see that ax is in execution stage and aw is in write back stage. Without data forwarding “ax” has to wait till “aw” completes write back. Hence, this is an instance to say that data forwarding is implemented.

3) The configuration of machine B is as follows (from the config_b.cfg file) is as follows:

- Total number of integer ALU’s = 1
- Total number of integer multipliers/dividers = 1
- Total number of integer Memory system ports to CPU = 1
- Total number of floating point ALU’s = 1
- Total number of floating point multipliers/dividers = 1
- Memory access bus width = 8 bytes
- Load/store queue size = 8
- Instruction Fetch queue size = 1
- Instruction decode width = 1 instruction/cycle.
- Instruction issue width = 1 instruction/cycle.
- Register update unit size= 1
- Machine follows out of order execution.
- Bimod type of branch predictor

IPC of the mcf program on machine B is 0.5263 (from sim_config_b.out file). As the IPC of machine B is greater than IPC of machine A, we can say that the performance of machine B is greater than machine A. We can also infer that executing instructions out of their order was the reason for such an increase in performance. With out of order execution, the instructions execute in different order to decrease the stall time. Because of this process, there may be a case where the result of a later instruction is calculated long before the results of previous instructions and there may be a chance of inputting this result wrongly into the previous instructions. So, commit stage is required to reorder the execution of instructions and decides when the results are to be stored after the execution process.

4) The configuration of machine C is as follows (from the config_c.cfg file) is as follows:

- Total number of integer ALU's = 4
- Total number of integer multipliers/dividers = 4
- Total number of integer Memory system ports to CPU = 4
- Total number of floating point ALU's = 4
- Total number of floating point multipliers/dividers = 4
- Memory access bus width = 8 bytes
- Load/store queue size = 8
- Instruction Fetch queue size = 1
- Instruction decode width = 1 instruction/cycle.
- Instruction issue width = 1 instruction/cycle.
- Register update unit size= 1
- Machine follows in order execution.
- Bimod type of branch predictor

IPC of the mcf program on machine C is 0.4879 (from sim_config_c.out file). As the IPC of machine A is same as machine C, therefore we can say that there is no increase in performance by increasing the number of resources.

5) The configuration of machine D is as follows (from the config_d.cfg file) is as follows:

- Total number of integer ALU's = 4
- Total number of integer multipliers/dividers = 4
- Total number of memory system ports to CPU = 4
- Total number of floating point ALU's = 4
- Total number of floating point multipliers/dividers = 4
- Memory access bus width = 8 bytes
- Load/store queue size = 8
- Instruction Fetch queue size = 1
- Instruction decode width = 1 instruction/cycle.
- Instruction issue width = 1 instruction/cycle.
- Register update unit size= 1
- Machine follows Out of order execution.
- Bimod type of branch predictor

IPC of the mcf program on machine D is 0.5263 (from sim_config_d.out file).

When number of memory ports = 4, issue width = 1 and decode width = 1 and all the other resources as above, then the IPC of mcf program on machine D is 0.5263. When number of memory ports = 1, issue width = 4 and decode width = 1 and all the other resources as above, then the IPC of mcf program on machine D is 0.6369. When number of memory ports = 1, issue width = 1 and decode width = 4 and all the other resources as above, then the IPC of mcf program on machine D is 0.5263. Therefore, we conclude that increasing the instruction issue width increases the performance. Yes, from the obtained results we can conclude that out of order execution is more effective in a wider machine with more resources.

6) The configuration of machine E is as follows (from the config_e.cfg file) is as follows:

- Total number of integer ALU's = 1
- Total number of integer multipliers/dividers = 1
- Total number of memory system ports to CPU = 1
- Total number of floating point ALU's = 1
- Total number of floating point multipliers/dividers = 1
- Memory access bus width = 8 bytes
- Load/store queue size = 8
- Instruction Fetch queue size = 4
- Instruction decode width = 4 instruction/cycle.
- Instruction issue width = 4 instruction/cycle.
- Register update unit size= 8
- Machine follows out of order execution.
- Bimod type of branch predictor

IPC of the mcf program on machine E is 0.6455 (from sim_config_e.out file). The instruction issue width, decode width and instruction fetch queue size of machine E is 4 times greater than machine B, while the rest of the configuration remaining the same. As the IPC of the mcf program on machine E is greater than machine B (i.e., 0.5263), we can say that the performance of machine E is greater than machine B. We can also say that the increase in the width is the reason for such an increase in performance.

7) The configuration of machine F is as follows (from the config_f.cfg file) is as follows:

- Total number of integer ALU's = 4
- Total number of integer multipliers/dividers = 4
- Total number of integer Memory system ports to CPU = 4
- Total number of floating point ALU's = 4
- Total number of floating point multipliers/dividers = 4
- Memory access bus width = 8 bytes
- Load/store queue size = 8
- Instruction Fetch queue size = 4
- Instruction decode width = 4 instruction/cycle.
- Instruction issue width = 4 instruction/cycle.
- Register update unit size= 8
- Machine follows out of order execution.
- Bimod type of branch predictor

IPC of the mcf program on machine F is 0.8697 (from sim_config_f.out file). The instruction issue width, decode width and instruction fetch queue size of machine C is 4 times greater than machine B. The register update unit size of machine F is 8 times than the register update unit size of machine A. The rest of the configuration remains the same. As the IPC of the mcf program on machine F is greater than machine C (i.e., 0.4879), we can say that the performance of machine F is greater than machine C. We can also say that the increase in the width of the machine is the has positive effects on performance.

8) The configuration of machine G is as follows (from the config_g.cfg file) is as follows:

- Total number of integer ALU's = 1
- Total number of integer multipliers/dividers = 1
- Total number of integer Memory system ports to CPU = 1
- Total number of floating point ALU's = 1
- Total number of floating point multipliers/dividers = 1
- Memory access bus width = 8 bytes
- Load/store queue size = 8
- Instruction Fetch queue size = 4
- Instruction decode width = 4 instruction/cycle.
- Instruction issue width = 4 instruction/cycle.
- Register update unit size= 16
- Machine follows out of order execution.
- Bimod type of branch predictor

IPC of the mcf program on machine G is 0.6752 (from sim_config_g.out file). The register update unit size of machine G is twice the register update unit size of machine E while the rest of the configuration remains the same as machine E. As the IPC of the mcf program on machine G is greater than machine E (i.e., 0.6455), we can say that the performance of machine G is greater than machine E. We can also say that the doubling the register update unit size is the reason for such an increase in performance.

9) The configuration of machine H is as follows (from the config_h.cfg file) is as follows:

- Total number of integer ALU's = 4
- Total number of integer multipliers/dividers = 4
- Total number of integer Memory system ports to CPU = 4
- Total number of floating point ALU's = 4
- Total number of floating point multipliers/dividers = 4
- Memory access bus width = 8 bytes
- Load/store queue size = 8
- Instruction Fetch queue size = 4
- Instruction decode width = 4 instruction/cycle.
- Instruction issue width = 4 instruction/cycle.
- Register update unit size= 16
- Machine follows out of order execution.
- Bimod type of branch predictor.

IPC of the mcf program on machine H is 1.0027 (from sim_config_h.out file). The number of functional units of machine H is twice the number of functional units of machine F, while the rest of the configuration remains the same as machine F. As the IPC of the mcf program on machine H is greater than machine F (i.e., 0.8697), we can say that the performance of machine H is greater than machine F. We can also say that the increasing the number of functional units is the reason for such an increase in performance.

10) Declaration: I have discussed this question with some of my classmates but made changes and run simulations on my own.

Branch mis-prediction rate:

Benchmark	Always Taken	Always Not-Taken	Bi-mod(2 bit)
Anagram	0.3126	0.6874	0.0386
test-math	0.3849	0.6151	0.1307
Bzip	0.4038	0.5962	0.0209
Gcc	0.3119	0.6881	0.0816
Mcf	0.4064	0.5936	0.0815

CPI of the corresponding branch predictor:

Benchmark	Always Taken	Always Not-Taken	Bi-mod(2 bit)
Anagram	0.6537	0.9619	0.4752
test-math	1.2105	1.3952	1.0788
Bzip	1.2966	0.9149	0.7712
Gcc	1.035	1.2052	0.9628
Mcf	1.1652	1.5597	1.004

Run time is strongly affected by the branch prediction rate. From my intuition, run time decreases if the branch prediction rate is high. From the above tables, if we consider a specific benchmark, the branch prediction rate is high for bi-mod type of prediction and thus, the CPI is lower for bi-mod as compared with other prediction schemes, that have lower prediction rates than bi-mod. As the CPI is lower it takes less time to execute a program. Thus, the conclusion of my results agree with my intuition.