ECE4700J Computer Architecture

Summer 2022

HW #1

Due: **2:59am (Beijing Time) June 6th, 2022**

*Please Submit a single **PDF** file on Canvas*

Q1 (10%): Please read the Turing lecture "A New Golden Age for Computer Architecture" offered by Prof. David Patterson and Prof. John Hennessy, and summarize **on your own words** at least **three** aspects about why it is "a golden age for computer architecture".

Sample Answers:

- DSA
- Hardware Security
- High-level Languages
- Open-source ISA such as RISC-V

Q2 (5%): Please list at least **two** reasons why CISC can be better over RISC.

Sample Answers:

- More efficient use of RAM than RISC
- The compiler has to do very little work to translate a high-level language statement into assembly.
- Small code sizes, high cycles per second
- …

The following table listed more details.

CISC	RISC
The original microprocessor ISA	Redesigned ISA that emerged in the early 1980s
Instructions can take several clock cycles	Single-cycle instructions
Hardware-centric design – the ISA does as much as possible using hardware circuitry	Software-centric design – High-level compilers take on most of the burden of coding many software steps from the programmer
More efficient use of RAM than RISC	Heavy use of RAM (can cause bottlenecks if RAM is limited)
Complex and variable length instructions	Simple, standardized instructions
May support microcode (micro-programming where instructions are treated like small programs)	Only one layer of instructions
Large number of instructions	Small number of fixed-length instructions
Compound addressing modes	Limited addressing modes

Source: https://www.microcontrollertips.com/risc-vs-cisc-architectures-one-better/

Q3 (20%): Assume the CPI of a processor is as follows:

- Loads = 6
- Stores = 5
- ALU operations = 4
- Conditional branches = 4
- Unconditional branches = 3

Architects are considering one of two enhancements. First, they are considering improving all ALU operations to be reduced to 3. Second, they are considering reducing all branches (conditional and unconditional) to 2. Using the **sjeng** benchmark from the table below, compute the average CPI before the enhancement and then with each enhancement and compute each enhancement's speedup over the original. The speedup is old CPI / new CPI. Redo this with the **mcf** benchmark. Compare the speedups between the two benchmarks and offer a suggestion of which speedup might be preferred.

Program	Loads	Stores	Branches	Jumps	ALU operations
astar	28%	6%	18%	2%	46%
bzip	20%	7%	11%	1%	54%
gcc	17%	23%	20%	4%	36%
gobmk	21%	12%	14%	2%	50%
h264ref	33%	14%	5%	2%	45%
hmmer	28%	9%	17%	0%	46%
libquantum	16%	6%	29%	0%	48%
mcf	35%	11%	24%	1%	29%
omnetpp	23%	15%	17%	7%	31%
perlbench	25%	14%	15%	7%	39%
sjeng	19%	7%	15%	3%	56%
xalancbmk	30%	8%	27%	3%	31%

Sample Answers:

sjeng original CPI = .19 * 6 + .07 * 5 + .56 * 4 + .15 * 4 + .03 * 3 = 4.42
sjeng enhancement 1 CPI = .19 * 6 + .07 * 5 + .56 * 3 + .15 * 4 + .03 * 3 = 3.86
sjeng enhancement 2 CPI = .19 * 6 + .07 * 5 + .56 * 3 + .15 * 2 + .03 * 2 = 4.09
speedup of enhancement 1 = 1.15 (15%)
speedup of enhancement 2 = 1.08 (8%)

mcf original CPI = .35 * 6 + .11 * 5 + .29 * 4 + .24 * 4 + .01 * 3 = 4.80
mcf enhancement 1 CPI = .35 * 6 + .11 * 5 + .29 * 3 + .24 * 4 + .01 * 3 = 4.51
mcf enhancement 2 CPI = .35 * 6 + .11 * 5 + .29 * 4 + .24 * 2 + .01 * 2 = 4.31
speedup of enhancement 1 = 1.06 (6%)
speedup of enhancement 2 = 1.11 (11%)

The two enhancements favor one benchmark over the other. Amdahl's law says to make the common case faster and in this case, the more common case is usually ALU operations (mcf is an exception in that the branches almost equal the ALU operations). So, even though the improvement in branches is better (4 -> 2, 3 -> 2 over 4 -> 3), the common case argues that enhancement 1 should be implemented.

Q4 (10%): Assume a new execution mode called "enhanced mode" provides a 2.5x speedup to the sections of programs where it applies. What percentage of a program (measured by original execution time) must run in enhanced mode for an overall speedup of 10%?

Sample Answers:

Let f be the required fraction.

$$\frac{T_{old}}{T_{new}} = 1.1$$

$$\frac{T_{old}}{\left((1-f)+\frac{f}{2.5}\right)\times T_{old}} = 1.1$$

$$\frac{1}{1-\frac{3f}{5}} = 1.1$$

$$\frac{3f}{5} = 1-\frac{1}{1.1}$$

$$f = 0.152\ or\ 15.2\%$$

Q5 (15%): Suppose a processor uses 105W of power while operating at 2.7 GHz, of which 3/4 is dynamic power. Suppose we want to run the same processor at a higher frequency which requires increasing the operating voltage proportionally as well.

1) (10%) What is the Power delay product? What is the Energy delay product? What is the Energy-delay^2 product?
2) (10%) If a dynamic power consumption increase of up to 130W can be tolerated, by how much can the processor frequency be sped up?

Sample Answers:

1)

PDP = Power * Delay = $\frac{1}{2.7\times10^9} = 3.89\times10^{-8}J$

EDP = PDP*t =$1.44\times 10^{-17}Js$

EDDP = EDP * t = $5.33\times 10^{-27}Js^2$

2)

$P \propto \alpha C V^2 f$

$\text{Speedup}^3 = \frac{130}{105 \times 3/4}$

$\text{Speedup} \approx 1.18$

Q6 (5%): Consider a 2.4 cm^2 die for a 64-bit processor manufactured from a 42 cm-diameter wafer costing $9,000. Assume a wafer yield of 99%. Use the defect model from the lecture notes with 0.016 defects per cm^2 and α=10. What is the expected cost per die (before testing)? Ignore edge effect correction.

Sample Answers:

The given equation for die yield is:

$$Die\ yield = Wafer\ yield \times \left(1 + (Defects\ per\ unit\ area \times Die\ area)\right)^{-\alpha}$$

Plugging in, we get:

$$Die\ yield = 0.99 \times \left(1 + (0.016 \times 2.4)\right)^{-10} = 0.679$$

Now we need to calculate the number of dies per wafer:

$$Dies\ per\ wafer = \frac{\pi \times \left(\frac{wafer\ diameter}{2}\right)^2}{Die\ area} - \frac{\pi \times wafer\ diameter}{\sqrt{2 \times Die\ area}}$$

$$Dies\ per\ wafer = \frac{\pi \times \left(\frac{42}{2}\right)^2}{2.4} - \frac{\pi \times 42}{\sqrt{2 \times 2.4}} = 577.27 - 60.23 = 517.04\ (rounded\ to\ 517)$$

Finally, we can figure out the cost per die based on the above calculations:

$$Cost\ per\ die = \frac{\$9000}{0.679 \times 517} \approx \$25.64$$

Q7 (10%): Consider a single-cycle datapath as a one-stage pipeline which runs at a frequency of 700MHz, if you want to break it into a S-stage pipeline. Assume that the clock overhead is 500ps, the probability of break b due to code delay is 0.2. If we assume that 5% of the additional cycle is used to address the quantization errors and other non-ideal effects. What would be the optimal number of stages for this pipeline? What is the new clock frequency? What is the throughput under this condition?

Sample Answers:

$$S_{opt} = \sqrt{\frac{(1-b)T \times 1.05}{bC}} = 3.47$$

S can be rounded to 3.

$$\Delta t = \frac{T}{S} + c \approx 1ns$$

$f \approx 1GHz$

$$\text{Throughput} = \frac{1}{\Delta t(1 + (S-1)b)} \approx 7.14 \times 10^8 IPS$$

Q8 (15%): Identify all data dependencies (potential data hazards) in the given code snippet. Assume the loop takes exactly one iteration to complete. Specify if the data dependence is RAW, WAW or WAR.

loop:

1. *ADDI R2, R2, #1*
2. *LD R4, 0(R3)*
3. *LD R5, 4(R3)*
4. *ADD R6, R4, R5*
5. *MUL R4, R6, R7*
6. *SUBI R3, R3, #8*
7. *BNEZ R2, loop*
8. *ADD R11, R12, R13*

Please answer in the following format.

Example:
1) 1 → 2 (RAW)

Sample Answers:

1) 1 → 2 (RAW)
2) 1→7 (RAW)
3) 2→4 (RAW)
4) 2→5 (WAW)
5) 2→6 (WAR)
6) 3→4 (RAW)
7) 3→6 (WAR)
8) 4→5 (RAW)
9) 4→5 (WAR)

Q9 (10%): The design of RISC-V provides for 32 general-purpose registers and 32 floating-point registers. If registers are good, are more registers better? List and discuss as many trade-offs (**at least 3**) as you can that should be considered by instruction set architecture designers examining whether to, and how much to, increase the number of RISC-V registers.

Sample Answers:

More registers are not necessarily better.

Pros:

- Less memory accesses
- Potential performance benefits
- Better handle of data dependency
- …

Cons:

- Area, power costs
- Context switching costs
- Longer instructions
- …