# Homework Assignment 11

CS/ECE 3810: Computer Organization
Nov 23, 2020

**Branch prediction, cache and virtual memory**

Due Date: Dec 2, 2020
(100 points)

Important Notes:

- Solutions turned in must be your own. Please, mention references (if any) at the end of each question. *Please refrain from cheating.*
- All solutions must be accompanied by the equations used/logic/intermediate steps. Writing only the final answer will receive **zero** credits.
- Partial score of every question is dedicated to each correct final answer provided by you. Please ensure both your equation/logic and final answer are correct. Moreover, you are expected to provide explanation for your solutions.
- All units must be mentioned wherever required.
- Late submissions **(after 11:59PM on 12/02/2020)** will not be accepted.
- We encourage all solutions to be typed in for which you could use software programs like LaTeX, Microsoft Word etc. If you submit handwritten solutions, they must be readable by the TAs to receive credits.

**Branch prediction.** A branch predictor tells us whether or not a branch is taken. By comparing this prediction and the actual outcome of the branch, we can compute the branch predictor's accuracy, which is given as:

$$\frac{\textit{Total number of correctly predicted outcomes}}{\textit{Total number of predictions}}$$

Consider a 2-bit branch predictor as discussed in Pipeline Hazards II lecture and Week-10 of the IVC meetings.

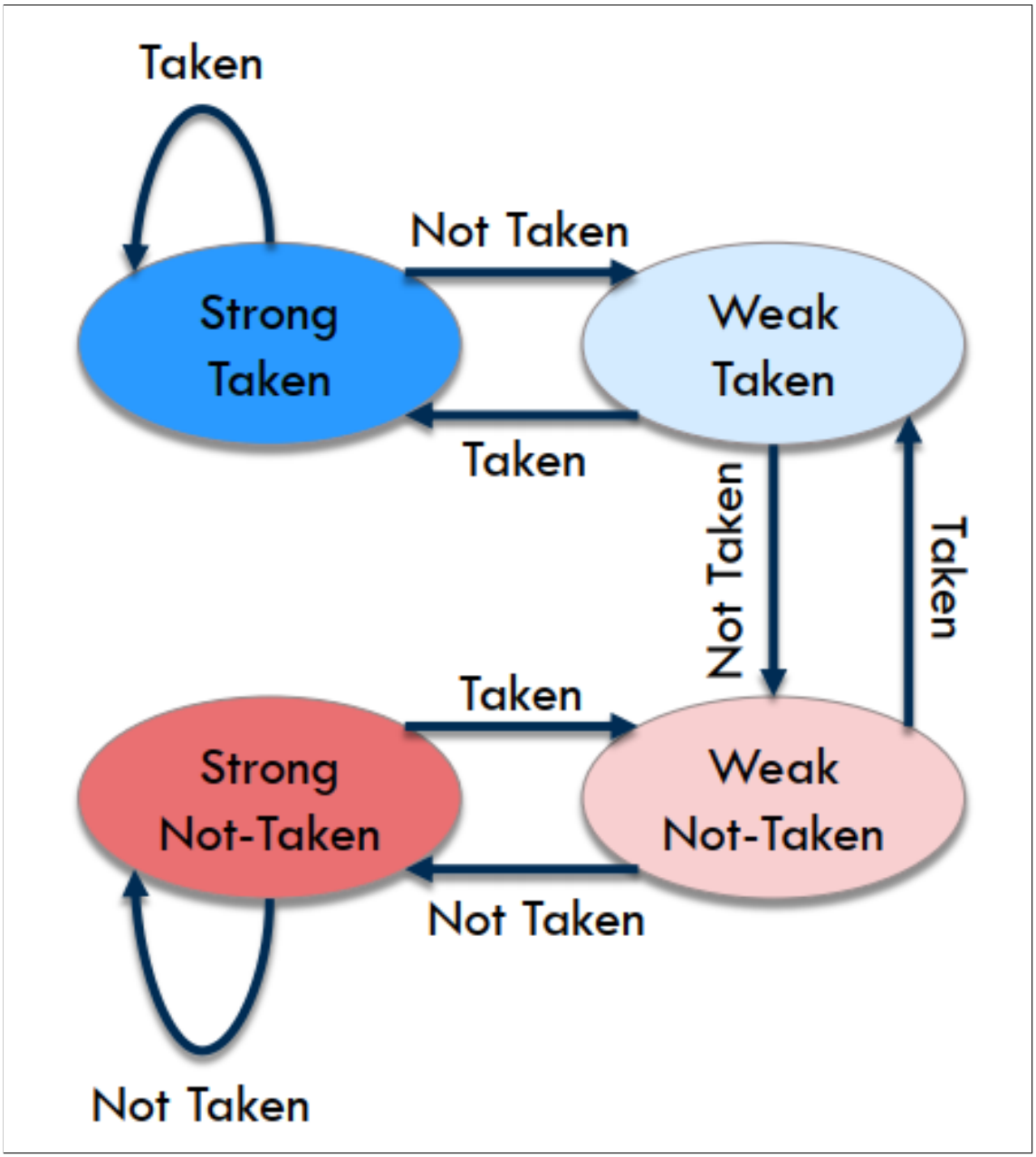


Figure 1: 2-bit branch predictor

If the branch predictor is in the *Strong Taken* or *Weak Taken* states, it predicts "Taken"; otherwise, it predicts "Not Taken". Given the following states, predictions and outcomes below,

| *State* | *Prediction* | *Outcome* | *Evaluation* |
|---|---|---|---|
| Strong Taken | Taken | Not Taken | Incorrect |
| Weak Taken | Taken | Taken | Correct |
| Strong Taken | Taken | Taken | Correct |
| Strong Taken | Taken | Not Taken | Incorrect |

We can evaluate the branch predictor's accuracy as $\frac{2}{4}$ or 50%.

**Question 1.** Consider we have a 2-bit branch predictor as discussed in the background section (Week-10 of the IVC meetings). For this branch predictor assume that the initial state is *Strong Taken*. All the branch outcomes are listed in the third column of the table below, following the state machine provided for this branch predictor fill out the table and also compute the accuracy of this branch predictor (Follow the same steps as explained in Week-10 of the IVC meetings). **(35 points)**

| *State* | *Prediction* | *Outcome* |
|---|---|---|
| Strong Taken | Taken | Not Taken |
| Weak Taken | Taken | Not Taken |
| Weak Not-Taken | Not Taken | Not Taken |
| Strong Not-Taken | Not Taken | Taken |
| Weak Not-Taken | Not Taken | Taken |
| Weak Taken | Taken | Not Taken |
| Weak Not-Taken | Not Taken | Taken |
| Weak Taken | Taken | Taken |

Accuracy $= \frac{total\ number\ of\ correctly\ predicted\ outcomes}{total\ number\ of\ predictions} = \frac{2}{8} = 25\%$

---

**Cache Miss Classifications.** Cache Miss occurs when data is not available in the cache. When the CPU detects a miss, it processes the miss by fetching the requested data from the main memory. Some of the ways to improve the cache hit rate are:
a) Prefetching,
b) Larger cache,
c) Larger cache block size, and
d) Greater set-associativity in the cache.

***Use the above options to answer the following question.***

**Question 2.** Recall the three classes of cache miss as,
a) Cold (compulsory)
b) Capacity
c) Conflict
from Slide #9 of the Cache III lecture.
Select which improvement(s) from the above is/are suitable for each class.
**(20 points)**

---

a) (a) (c)
Cold (compulsory) misses can be improved by prefetching and larger cache block size.

b) (b)
Capacity misses can be improved by larger cache.

c) (b) (d)
Conflict misses can be improved by larger cache and greater set-associativity in the cache.

**Cache Replacement Policies.** In microprocessors, cache replacement algorithms (also cache replacement policies) can improve the average memory access time (AMAT) by increasing the number of cache hits. Therefore, a good cache replacement policy can reduce the miss rate, thereby lowering the average memory access time (AMAT), given by the following equation.

$$\text{AMAT} = (t_h) + (r_m)*(t_p)$$

where $(t_h)$ is the hit time, $(t_p)$ is the miss penalty, and $(r_m)$ is the miss rate.

**Question 3.** Consider a 2-way set-associative cache with a single set. The following access pattern is observed when running a program.

*A, B, D, A, B, C, D, C, A*

Compute the miss rates of the IDEAL, LRU, and MRU replacement policies for this access pattern. Refer to slides #18 - #26 of the Cache III lecture.**(30 points)**

| *Policy* | *A* | *B* | *D* | *A* | *B* | *C* | *D* | *C* | *A* | *Miss Rate* |
|---|---|---|---|---|---|---|---|---|---|---|
| *IDEAL* | X | X | X | X | ✓ | X | X | X | X | 8/9 |
| *LRU* | X | X | X | X | X | X | X | ✓ | X | 8/9 |
| *MRU* | X | X | X | ✓ | X | X | X | X | X | 8/9 |

---

**Virtual Memory.** It is an integral part of modern computer architecture; implementations usually require hardware support, typically in the form of a memory management unit built into the CPU. It creates an illusion of a very large (main) memory, by providing an abstraction of the resources available on a given machine.

**Question 4.** Assume that we have a memory system detailed as follows:
- 4KB page size
- 256MB Physical Address Space
- 4GB Virtual Address Space

a) How many bits will be used for the page-offset?
b) How many bits will be used for the physical page number (PPN)?
c) How many entries are there in the page table?
**(15 points)**

---

a) For a 4KB page size, the page-offset is 12 bits.

b) The bits for the PPN are: 32 - 12 (offset) = 20.

c) Using the value from (b) the number of entries in the page table is $2^{20}$