

國立中山大學 114 學年度 碩士班考試入學招生考試試題

科目名稱：計算機結構【資工系碩士班甲組、乙組】

—作答注意事項—

考試時間：100 分鐘

- 考試開始鈴響前不得翻閱試題，並不得書寫、劃記、作答。請先檢查答案卷（卡）之應考證號碼、桌角號碼、應試科目是否正確，如有不同立即請監試人員處理。
- 答案卷限用藍、黑色筆(含鉛筆)書寫、繪圖或標示，可攜帶橡皮擦、無色透明無文字墊板、尺規、修正液（帶）、手錶(未附計算器者)。每人每節限使用一份答案卷，請衡酌作答。
- 答案卡請以 2B 鉛筆劃記，不可使用修正液（帶）塗改，未使用 2B 鉛筆、劃記太輕或污損致光學閱讀機無法辨識答案者，後果由考生自負。
- 答案卷（卡）應保持清潔完整，不得折疊、破壞或塗改應考證號碼及條碼，亦不得書寫考生姓名、應考證號碼或與答案無關之任何文字或符號。
- 可否使用計算機請依試題資訊內標註為準，如「可以」使用，廠牌、功能不拘，唯不得攜帶書籍、紙張（應考證不得做計算紙書寫）、具有通訊、記憶、傳輸或收發等功能之相關電子產品或其他有礙試場安寧、考試公平之各類器材入場。
- 試題及答案卷（卡）請務必繳回，未繳回者該科成績以零分計算。
- 試題採雙面列印，考生應注意試題頁數確實作答。
- 違規者依本校招生考試試場規則及違規處理辦法處理。

國立中山大學 114 學年度碩士班考試入學招生考試試題

科目名稱：計算機結構【資工系碩士班甲組、乙組】

題號：434001

※本科目依簡章規定「不可以」使用計算機(問答申論題)

共 2 頁第 1 頁

1. (18%) A memory system is composed of **eight banks**, and each bank contains 2^{16} rows. Every DRAM row refresh is initiated by a command from the memory controller, and it refreshes a single row in a single DRAM bank. Each refresh command keeps the command bus busy for **5ns**. We define **command bus utilization** as the fraction of total execution time during which the command bus is occupied.
- 1.1 (6%) Given that the refresh interval is **64ms**, calculate the command bus utilization of refresh commands. Show your work step-by-step.
- 1.2 (12%) If **70%** of all rows can withstand a refresh interval of **256ms**, how does the command bus utilization of refresh command change? Calculate the reduction $(1 - \frac{new}{old})$ in bus utilization. Show your work step-by-step.
2. (14%) Assume we have a virtual memory detailed as follows:
- 256 MiB Physical Address Space
 - 4 GiB Virtual Address Space
 - 1 KiB page size
 - A TLB with 4 sets that is 8-way associative with LRU replacement
- 2.1 (10%) How many bits will be used for **page offset**, **Virtual Page Number (VPN)**, **Physical Page Number (PPN)**, **TLB index**, and **TLB tag**?
- 2.2 (4%) How many entries are in this page table?
3. (18%) Assume a machine with a **7-stage** pipeline. Assume that branches are resolved in the sixth stage. Assume that 20% of instructions are branches.
- 3.1 (4%) How many instructions of wasted work are there per branch misprediction on this machine?
- 3.2 (4%) Assume **N** instructions are on the correct path of a program and assume a branch predictor accuracy of **A**. Write the equation for the number of instructions that are fetched on this machine in terms of **N** and **A**. (Show your work step by step)
- 3.3 (4%) Let's say we modify the machine so that it uses dual path execution (where an equal number of instructions are fetched from each of the two branch paths). Assume branches are resolved before new branches are fetched. Write how many instructions would be fetched in this case, as a function of **N**. (Show your work step by step)
- 3.4 (6%) Now let's say that the machine combines branch prediction and dual path execution in the following way: A branch confidence estimator is used to gauge how confident the machine is of the prediction made for a branch. When confidence in a prediction is high, the branch predictor's prediction is used to fetch the next instruction; when confidence in a prediction is low, dual path execution is used instead. Assume that the confidence estimator estimates a fraction **C** of the branch predictions have high confidence, and that the probability that the confidence estimator is wrong in its high confidence estimation is **M**. Write how many instructions would be fetched in this case, as a function of **N**, **A**, **C**, and **M**. (Show your work step by step)
4. (14%) Suppose that (after optimization) a typical **n**-instruction program requires an additional **4*n NOP** instructions to correctly handle data hazards.
- 4.1 (4%) Suppose that the cycle time of this pipeline without forwarding is 250 ps. Suppose also that adding forwarding hardware will reduce the number of **NOPs** from **0.4*n** to **0.05*n**, but increase the cycle time to 300 ps. What is the speedup of this new pipeline compared to the one without forwarding?
- 4.2 (4%) Can a program with only **0.075*n NOPs** possibly run faster on the pipeline with forwarding? Explain why or why not.
- 4.3. (6%) At minimum, how many **NOPs** (as a percentage of code instructions) must a program have before it can possibly run faster on the pipeline with forwarding?

試題請隨卷繳回，請留意背面是否有題

國立中山大學 114 學年度碩士班考試入學招生考試試題

科目名稱：計算機結構【資工系碩士班甲組、乙組】

題號：434001

※本科目依簡章規定「不可以」使用計算機(問答申論題)

共 2 頁第 2 頁

5. (18%) Consider a single-issue in-order multithreading processor. Each cycle, the processor can fetch and issue one instruction that performs any of the following operations:

- load/store, 13-cycle latency (fully pipelined)
- integer add, 1-cycle latency
- floating-point add, 6-cycle latency (fully pipelined)
- branch, no delay slots, 1-cycle latency

The processor **does not have a cache**. Each memory operation directly accesses main memory. If an instruction cannot be issued due to a data dependency, the processor stalls. We also assume that the processor has a perfect branch predictor with no penalty for both taken and not-taken branches. Please analyze the processor utilizations for the following two thread-switching implementations:

Fixed Switching: the processor switches to a different thread every cycle using fixed round robin scheduling. Each of the N threads executes an instruction every N cycles.

Data-dependent Switching: the processor only switches to a different thread when an instruction cannot execute due to a data dependency.

Each thread executes the following RISC-V code:

```
loop:  LD      f2,      0(x1)      # load data into f2
      ADDI    x1,      x1,      4      # bump pointer
      FADD    f3,      f3,      f2      # f3 = f3 + f2
      BNE     f2,      f4,      loop    # continue if f2 != f4
```

5.1 (8%) What is the minimum number of threads that we need to fully utilize the processor for each thread-switching implementation? Explain your reasoning.

5.2 (10%) What is the minimum number of threads that we need to fully utilize the processor for each thread-switching implementation if we change the **load/store latency to 1-cycle (but keep the 6-cycle floating-point add)**? Explain your reasoning.

6. (18%) Processor microarchitecture : VLIW vs. Superscalar vs. Array Processor

6.1 (3%) What do VLIW, superscalar execution, and array processing concepts have in common?

6.2 (6%) Provide two reasons why a VLIW microarchitecture is simpler than a "same-width" superscalar microarchitecture.

6.3 (3%) Provide a reason why a superscalar microarchitecture could provide higher performance than a "same-width" VLIW microarchitecture.

6.4 (3%) Provide a reason why a VLIW processor is more flexible than an array processor.

6.5 (3%) Provide a reason why an array processor is simpler than a "same-width" VLIW processor.