Project 1

Kumaravel Jagasivamani
Mar. 27, 2024

Requirements

Input Buffer

Address Offset	Sample	Value
0	0	X
8	0	Y
16	0	Z
24	0	O
32	1	X
40	1	Y
48	1	Z
56	1	O
64	2	X
72	2	Y
80	2	Z
88	2	O

Requirements

X,Y,Z,O buffer

Address Offset	Sample
0	0
8	1
16	2

Pseudo-code

- Perform for each sample:
  - Read X of current sample from input_buffer
  - Write X of current sample to X buffer
  - Read Y of current sample from input_buffer
  - Write Y of current sample to Y buffer
  - Read Z of current sample from input_buffer
  - Write Z of current sample to Z buffer
  - Read O of current sample from input_buffer
  - Write O of current sample to O buffer

Registers and Data Memory

- Registers:
  - rax, rbx, rdi, rsi, rcx, rdx, …
  - Refers to content of register (fixed location in processor)
- Data Memory:
  - Defined in .bss section
    - loop_index resq 1 ; Quad-word (8 bytes)
  - loop_index: address of the value
  - [loop_index]: content of memory location (value read/written)

Pseudo-code

- For ([loop_index]=0;[loop_index]<[samples_cnt];[loop_index]++):
  - Read X from input_buffer
  - Write X to X buffer
  - Read Y from input_buffer
  - Write Y to Y buffer
  - Read Z from input_buffer
  - Write Z to Z buffer
  - Read O from input_buffer
  - Write O to O buffer

Pseudo-code

Separate loops

- For ([loop_index]=0;[loop_index]<[samples_cnt];[loop_index]++):
  - Read X from input_buffer
  - Write X to X buffer
- For ([loop_index]=0;[loop_index]<[samples_cnt];[loop_index]++):
  - Read Y from input_buffer
  - Write Y to Y buffer
- For ([loop_index]=0;[loop_index]<[samples_cnt];[loop_index]++):
  - Read Z from input_buffer
  - Write Z to Z buffer
- For ([loop_index]=0;[loop_index]<[samples_cnt];[loop_index]++):
  - Read O from input_buffer
  - Write O to O buffer

Loop

- start_loop_label:
  - <statements in loop>
  - Increment [loop_index]
  - Compare [loop_index] with [samples_cnt]
  - If not equal, jump back to start_loop_label
  - If equal, exit loop

Read X from input_buffer

- Sample 0:
  - [input_buffer + 0]
- Sample 1:
  - [input_buffer + 32]
- Sample 2:
  - [input_buffer + 64]
- input_buffer: starting memory address of data buffer
- [input_buffer+32]: value starting at 32 byte offset from input_buffer (sample 1’s x)

Read X from input_buffer

- Sample 0:
  - [input_buffer + [loop_index]*32]
- Sample 1:
  - [input_buffer + [loop_index]*32]
- Sample 2:
  - [input_buffer + [loop_index]*32]
- input_buffer: starting memory address
- loop_index: address of memory location containing loop index
- [loop_index]: actual value in memory location

Read X from input_buffer

- rdi = input_buffer
- Sample 0:
  - [rdi + [loop_index]*32]
- Sample 1:
  - [rdi + [loop_index]*32]
- Sample 2:
  - [rdi + [loop_index]*32]

Read X from input_buffer

- rdi: input_buffer
- rax: [loop_index]
- rax = rax*32
- rdi = rdi + rax
- rbx = [rdi]

Write X to X buffer

- Sample 0:
  - [x + 0]
- Sample 1:
  - [x + 8]
- Sample 2:
  - [x + 16]

Write X to X buffer

- Sample 0:
  - [x +[loop_index]*8]
- Sample 1:
  - [x + [loop_index]*8]
- Sample 2:
  - [x + [loop_index]*8]

Write X to X buffer

- rsi = x
- rax = [loop_index]
- rax = rax*8
- rsi = rsi + rax
- [rsi] = rbx
-

Read Y from input_buffer

- rdi: input_buffer
- rax: [loop_index]
- rax = rax*32
- **rax = rax + 8**
- rdi = rdi + rax
- rbx = [rdi]

Write Y to Y buffer

- rsi = y
- rax = [loop_index]
- rax = rax*8
- rsi = rsi + rax
- [rsi] = rbx
-

Checking design

- Change the number of samples in cmsc313_proj1.c:
  - const int data[] = {53, 33, 38, 85, 153, 133, 138, 185, 253, 233, 238, 585};
  - const int data[] = {53, 33, 38, 85, 153, 133, 138, 185, 253, 233, 238, 585**, 83, 283, 38, 832**};
  - const int data[] = {53, 33, 38, 85, 153, 133, 138, 185, 253, 233, 238, 585**, 83, 283, 38, 832, 765, 294, 173, 472**};
- Change the output being displayed (change o to x,y,z to check each of them):
  - ; Start of loop to print values in o array
  - print_loop:
  - ; rdi is the pointer to the o array
  - mov rdi, **o**

Optimizations

- Increment input_buffer offset after each read operation
  - No need to use [loop_index]
  - If rdi stores the input_buffer offset, add 8 to rdi
- Reuse same offset value for x,y,z,o buffers
  - For sample x, the offset from the base of the buffer is the same
  - Dedicate a register to calculate the offset for x
  - Reuse the register for y,z,o
- Load separate registers for base address of x,y,z,o buffer
  - Don’t need to reuse rsi for all output buffers
  - Registers that could be used: rcx, rdx, r8, …, r15