ECE/CS 252
Intro to Computer Engineering

Week 08 Discussion

1

Example Load Instruction

Control						
PC	0x3FB2	N	1	Z	0	P 0
IR	0110 0111 0011 1100					

Register File			
R0	0x3FB5	R4	0x3FC8
R1	0x9F08	R5	0x0000
R2	0xFFFF	R6	0x8000
R3	0xCAFE	R7	0x4242

Memory	
0x3FBD	0x9876
0x3FBE	0x2397
0x3FBF	0x6014
0x3FC0	0x3FC9
0x3FC1	0xFC00
0x3FC2	0x3FBE
0x3FC3	0x9944
0x3FC4	0x1234
0x3FC5	0x00FF
0x3FC6	0xAAAA

2

Example Store Instruction 1

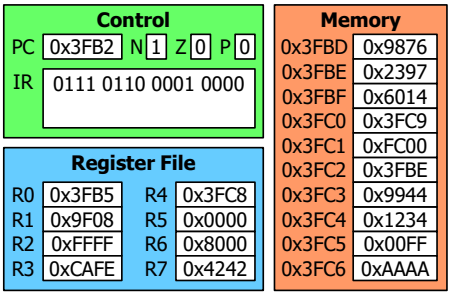
Control						
PC	0x3FB2	N	1	Z	0	P 0
IR	0011 0110 0001 0000					

Register File			
R0	0x3FB5	R4	0x3FC8
R1	0x9F08	R5	0x0000
R2	0xFFFF	R6	0x8000
R3	0xCAFE	R7	0x4242

Memory	
0x3FBD	0x9876
0x3FBE	0x2397
0x3FBF	0x6014
0x3FC0	0x3FC9
0x3FC1	0xFC00
0x3FC2	0x3FBE
0x3FC3	0x9944
0x3FC4	0x1234
0x3FC5	0x00FF
0x3FC6	0xAAAA

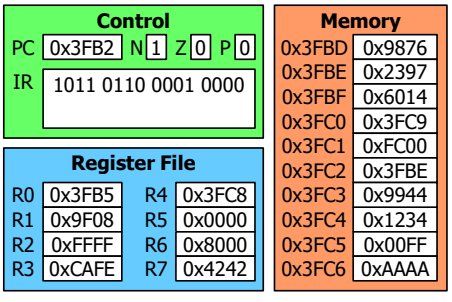
3

Example Store Instruction 2



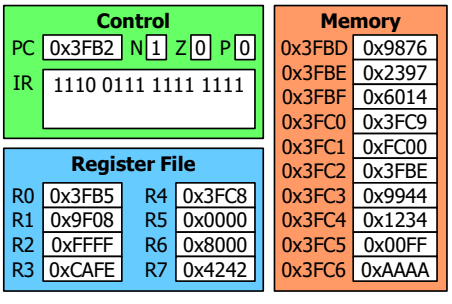
4

Example Store Instruction 3



5

Example Data Movement Instr.



6

Which of these do R1←R0?

```
AND   R1, R0, x1
ADD   R1, R0, x0
LD    R1, R0
LD    R1, x0
LDR   R1, R0, x0
```

7

Write a program to do A[0] = A[1] – A[2]

```
        .ORIG x0200
START

        BR START        ; repeat the program

A       .FILL  x42      ; start of the array
        .FILL  b111
        .FILL  #-3
        .END
```

8

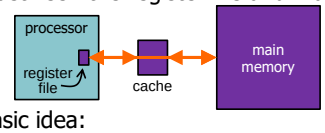
Processor Memory Systems

- In real processor systems, accessing registers is MUCH faster than accessing memory
  - Not a small difference – it's orders of magnitude!
- The register file is relatively fast because it is small and inside the processor core
  - Unfortunately, fast == large area and high power
- The main memory is slow because of its size, technology, and distance from the processor core
- So, in programs we try very hard to NOT access memory unless we have to!

9

Tiered Memory Systems

- To reduce memory access time, put intermediate stage between the register file and main memory



- The basic idea:
  - Create a small/fast cache memory between the register file and main memory to hold recently accessed data
  - Memory load: If the data isn't already in the cache, copy a block of locations that includes it to the cache
  - Memory store: If it is in the cache, update only the cache but mark it as "dirty" (write back when have to)
  - Automatically manage the cache memory in hardware

Cache Memory

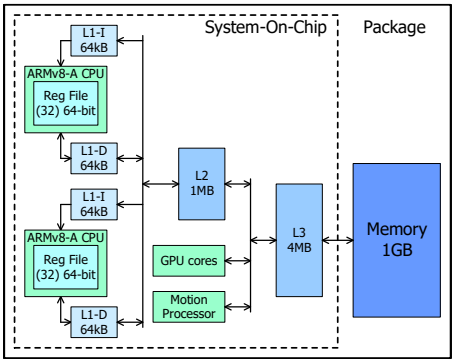
- This only works because programs don't access memory randomly
  - **Spatial locality**: accessing a memory address probably means that ones nearby will be accessed soon
  - **Temporal locality**: accessing a memory address probably means that it will be accessed again soon
- The cache is smaller than main memory, so it can only hold a subset of everything in memory
  - The trick is to try to have the best subset
  - Once the cache is full, when a new location is accessed we need to kick something out of the cache!

Cache Memory Hierarchy

- If one cache is a good idea, let's add more!
- Progressively larger caches further from processor
- Split the first level cache into separate instruction and data caches
  - The pattern of instruction fetches is very different from the common patterns of data accesses
  - Instructions are usually fetched sequentially in blocks, and we only read those locations
  - Data is often read and written from sequences of regularly spaced locations
  - Split caches let us optimize the I-cache and D-cache separately based on their different access patterns

Apple A7 SoC (iPhone 5S)

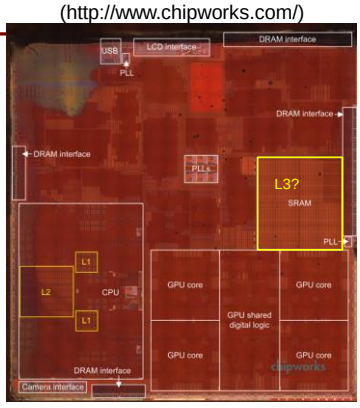
(approximately…)



13

Die Photo!

photo from Chipworks
(http://www.chipworks.com/)



14

Wrapping Up

- Up Next:
  - LC-3 Control Instructions and Programming
  - Programming/Debugging Techniques
- Remember your videos and reading
  - Including the video quiz!
- Questions?

15

16

17