CS 451 / 551 / ECE 541

ADVANCED COMPUTER ARCHITECTURE

SESSION no. 10

University of Idaho

PROJECT PROPOSALS

IN CLASS: DUE TUESDAY OCT 8
EO: .. THURSDAY OCT 10

Email: [class_Proposal.name.{pdf/word}
ECE541

1-2 pages
- Research project
- Programming / Implementation / Design

what will you research or implement?
- why is it important?
- what will you learn?
- what sources/resources will you use?

University of Idaho

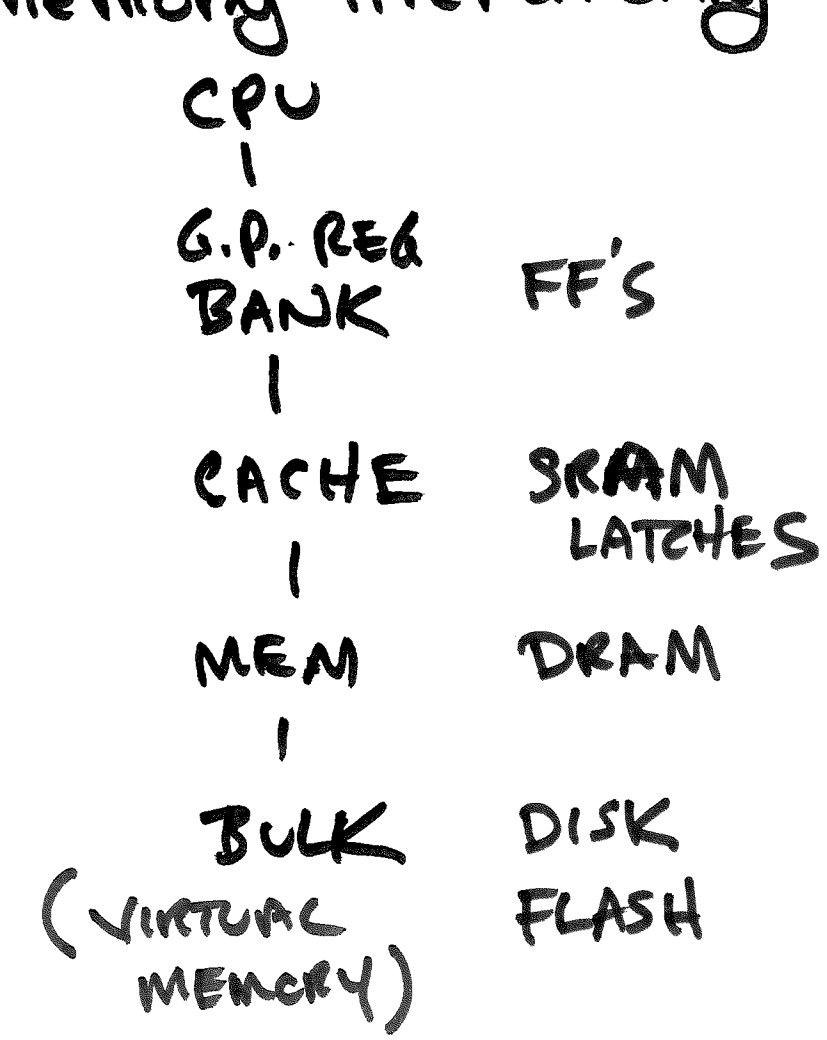
RESEARCH

- Explore a topic in depth
- Emerging trend

IMPLEMENTATION

- Best path to thorough understanding
- Many questions will arise!

Memory Hierarchy



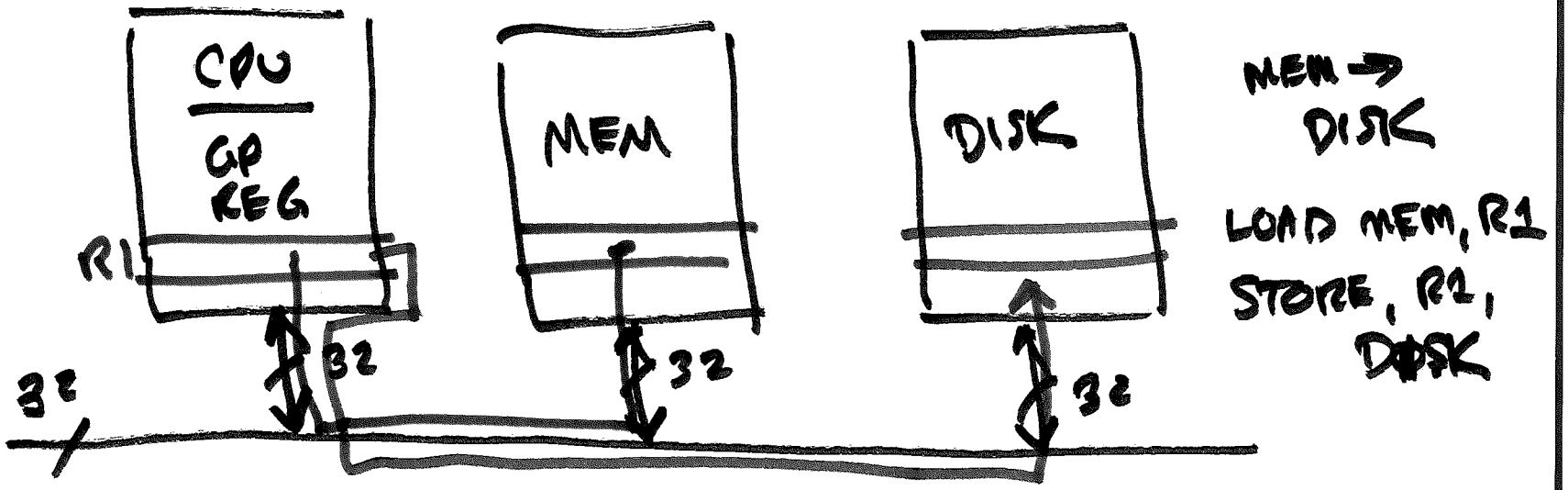
ILLUSION: FAST, CHEAP, LARGE CAPACITY.

CACHE -

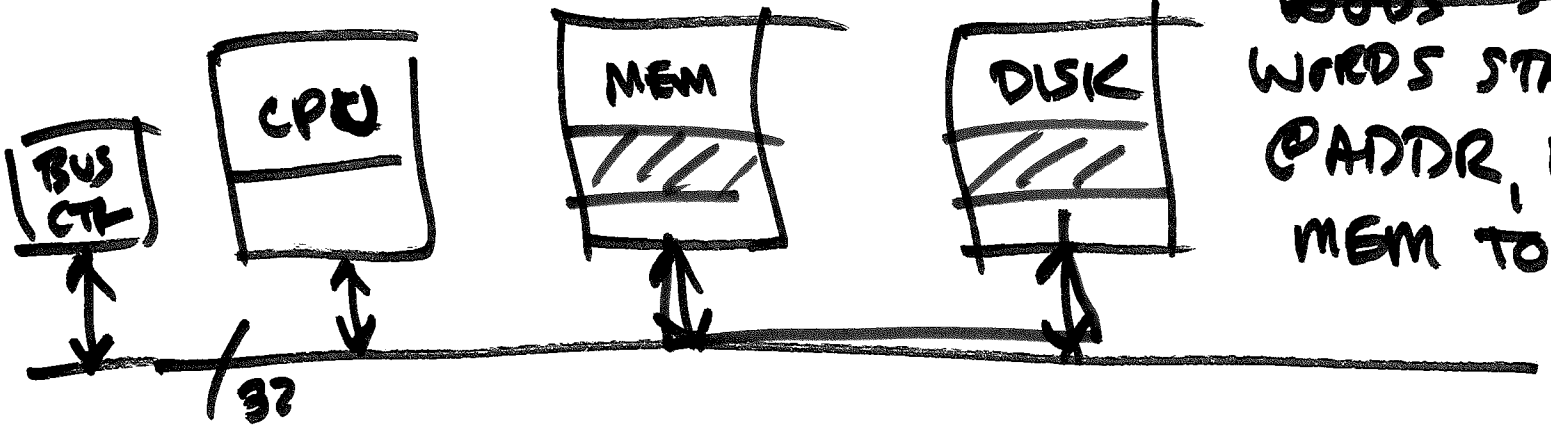
- local, fast, temporary storage.
- Window on main memory

PREFER BLOCK TRANSFERS -

University of Idaho



WITH DMA



TRANSFER 512 ~~WORDS STARTING~~ WORDS STARTING @ADDR, FROM MEM TO DISK.

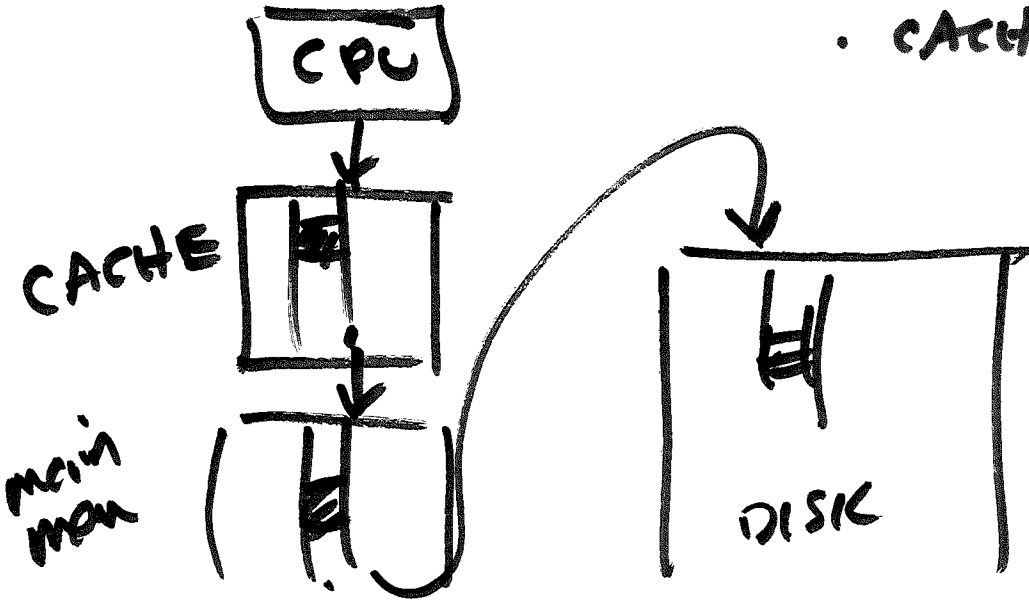
University of Idaho

EXTEND IDEA -

VIRTUAL MEMORY

e.g. 32 bit address space => 4 GB
(byte addressable)

- Have 1/2 GB of DRAM
- Use 3 1/2 GB on disk, as if it were DRAM



- CACHE - negotiations in hardware
- VM - negotiation in SW - O/S.
- Transparent to programmer!

CACHE

Cache "block" - window on mem

Locality of reference

- Spatial
- Temporal

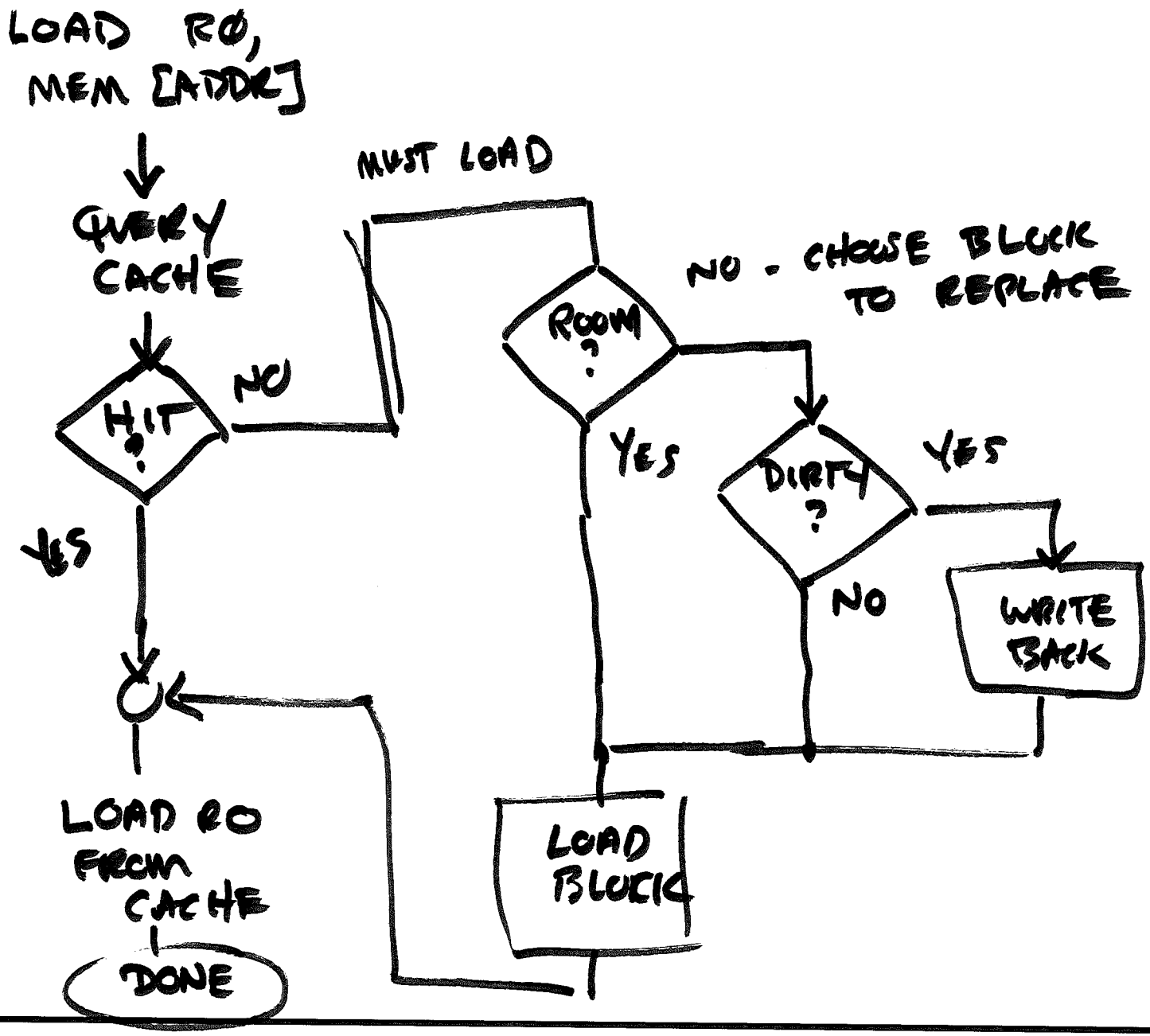
FIRST - OBSERVATION

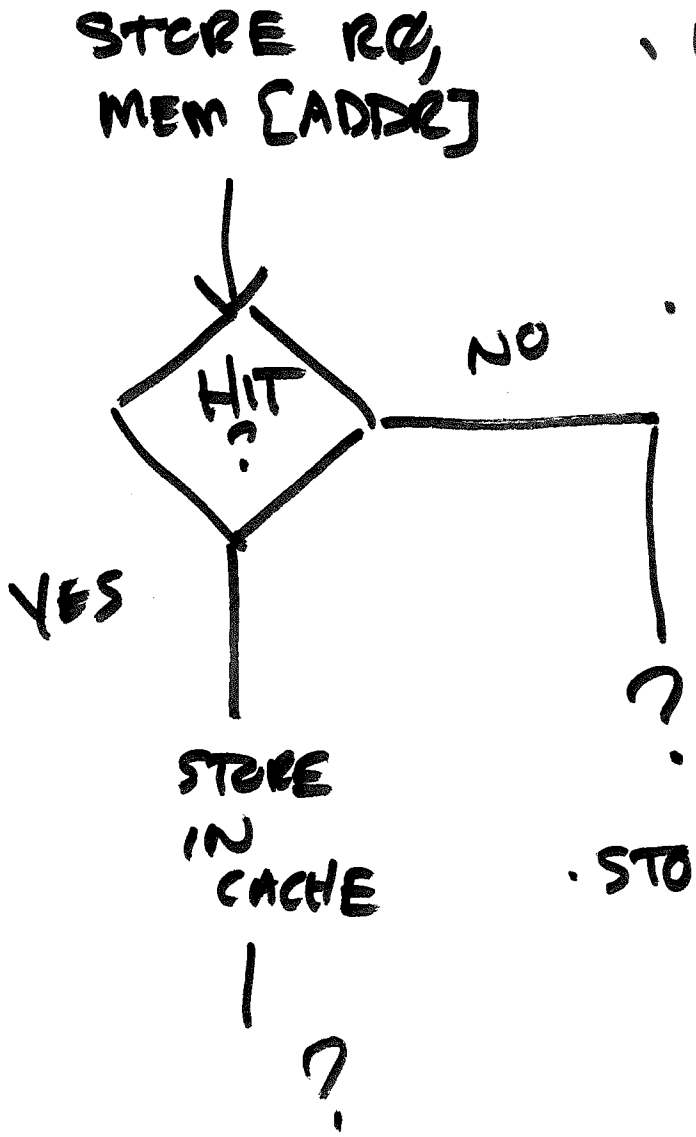
SECOND - ENFORCE LOCALITY

- Modular code

Different apps. have different patterns

- SPEC benchmarks
- C → C++ → Java / JVM → Python

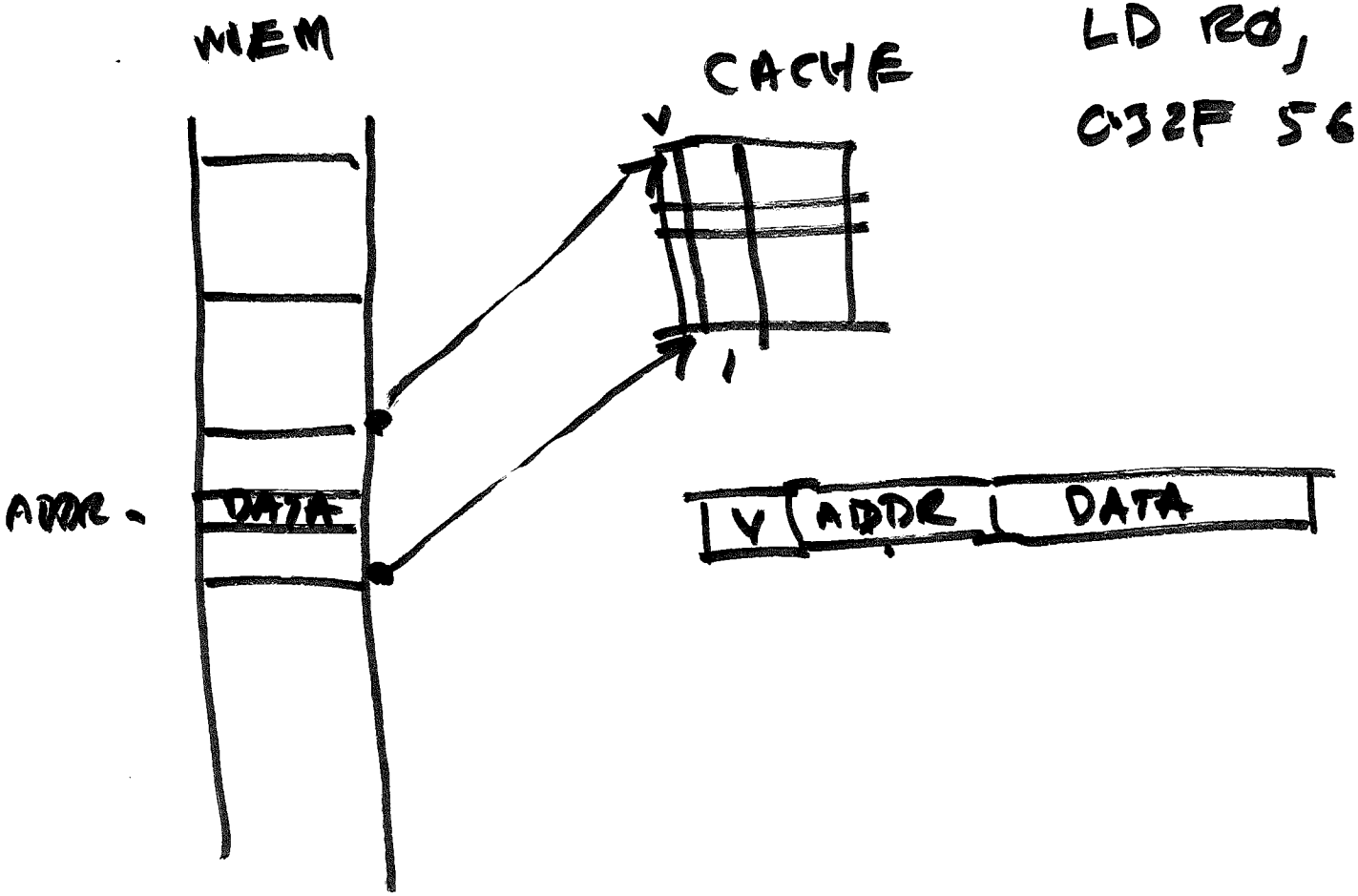

LOAD R0, MEM [ADDR]
QUERY CACHE
HIT?
NO
MUST LOAD
ROOM?
NO - CHOOSE BLOCK TO REPLACE
YES
DIRTY?
YES
NO
WRITE BACK
YES
LOAD R0 FROM CACHE
LOAD BLOCK
DONE



- WRITE THROUGH - ALWAYS WRITE TO CACHE & MEM
- WRITE BACK - ONLY WRITE TO CACHE
  - IF MISS — WRITE BLOCK TO MEM.

- STORE IN MAIN MEM

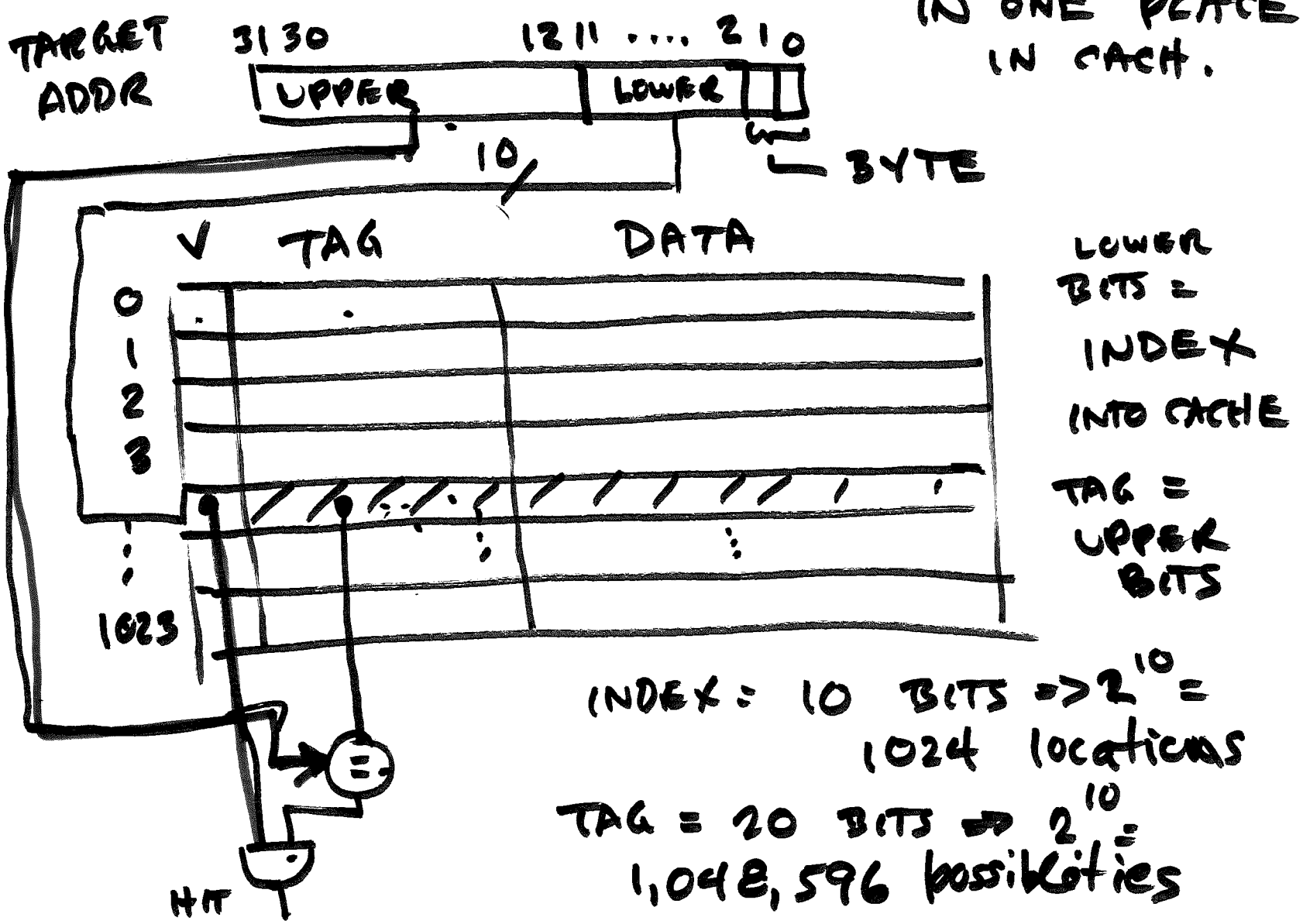
DIRECT-MAPPED



University of Idaho

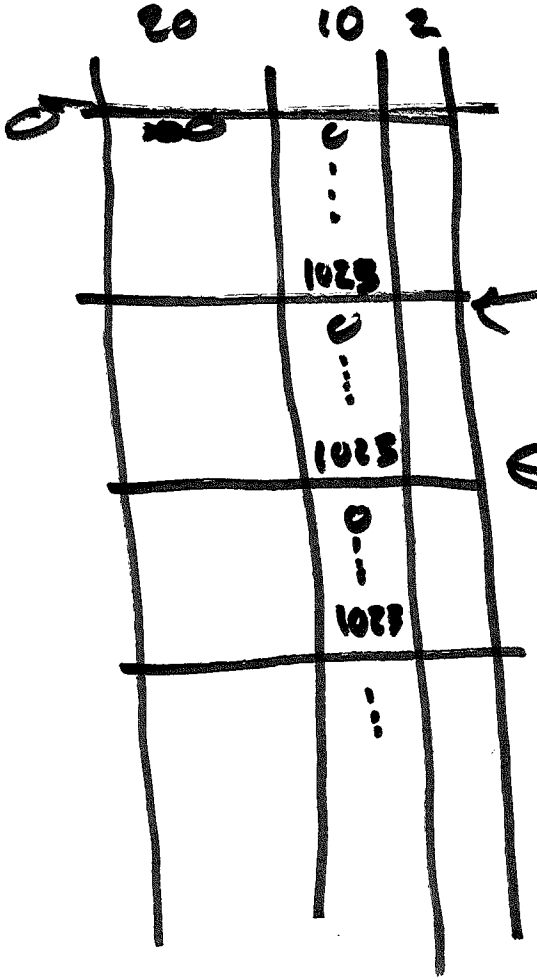
DIRECT-MAPPED

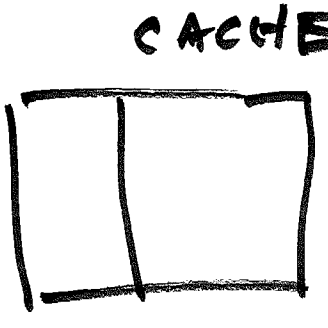
A MEMORY BLOCK CAN ONLY GO IN ONE PLACE IN CACH.

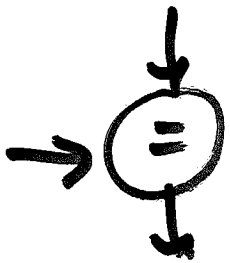


INDEX: 10 BITS $\Rightarrow 2^{10}$ = 1024 locations

TAG = 20 BITS $\Rightarrow 2^{10}$ = 1,048,596 possibilities

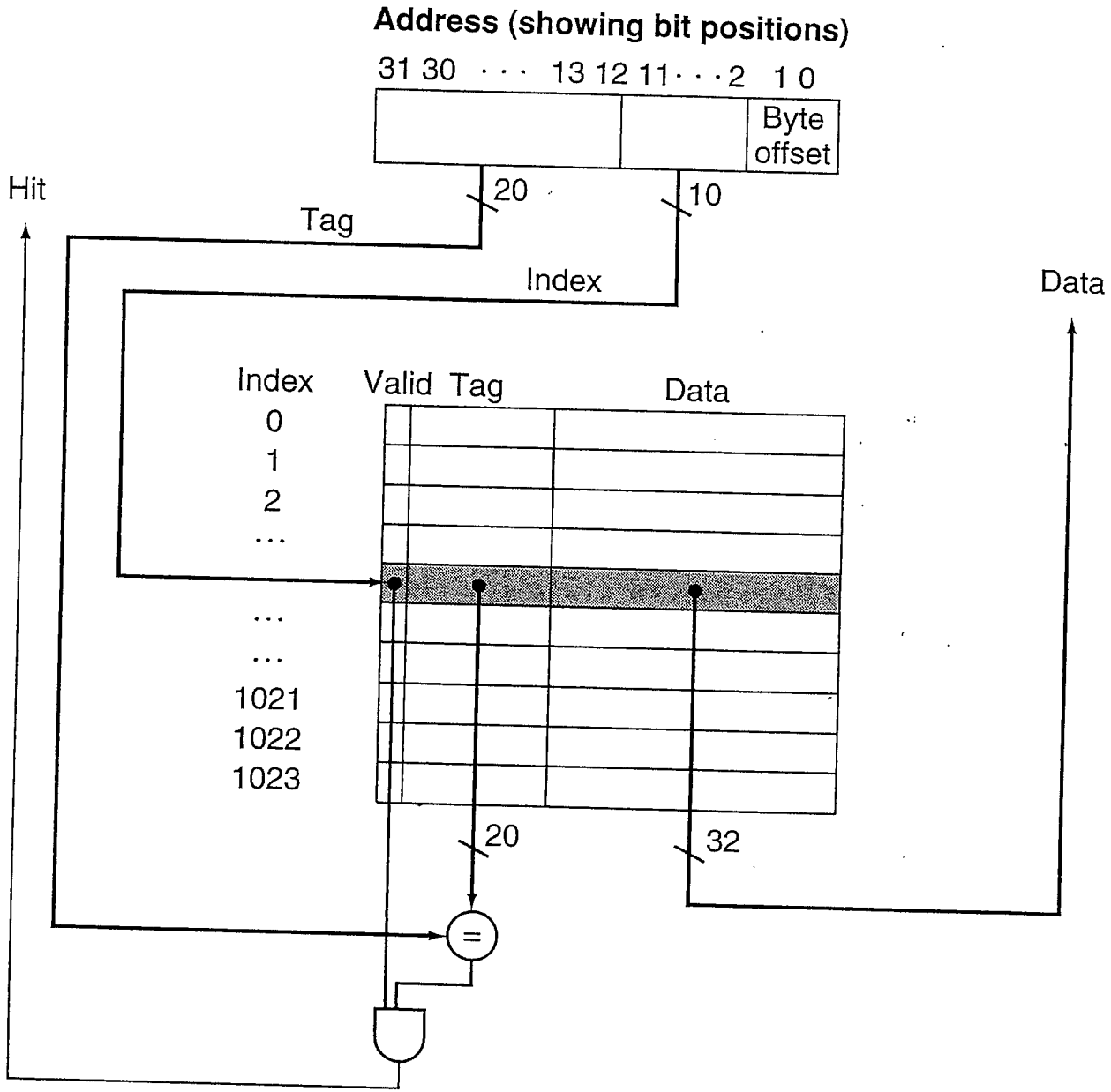






COMPARATOR

A	B	A⊕B	$\overline{A \oplus B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1


Address (showing bit positions)
31 30 · · · 13 12 11 · · · 2 1 0
Byte offset
20
10
Tag
Index
Hit
Data
Index
Valid
Tag
Data
0
1
2
· · ·
· · ·
· · ·
1021
1022
1023
20
32
=

DIRECT MAPPED - SIMPLE MATCHING HARDWARE. MEMORY ELEMENT CAN ONLY BE IN ONE LOCATION (INDEX)

CACHE MISSES - ON I.F.

. Pipelined processor

1. Back up PC - point to instr not in cache
2. Command memory read - load block to cache.
3. Set valid bit
4. Restart instruction - Now instr is in cache (Hit)

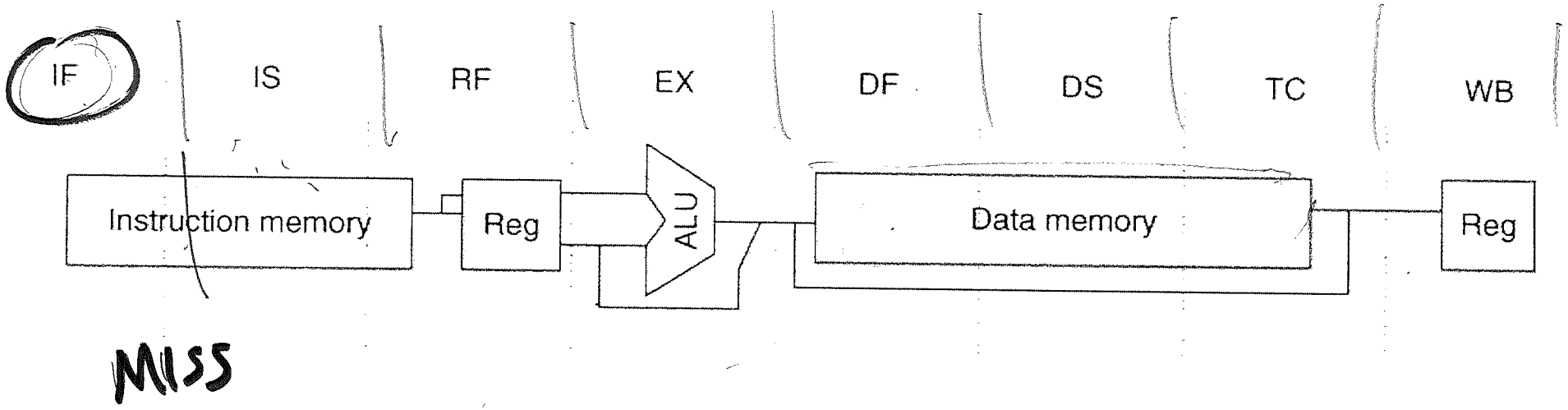


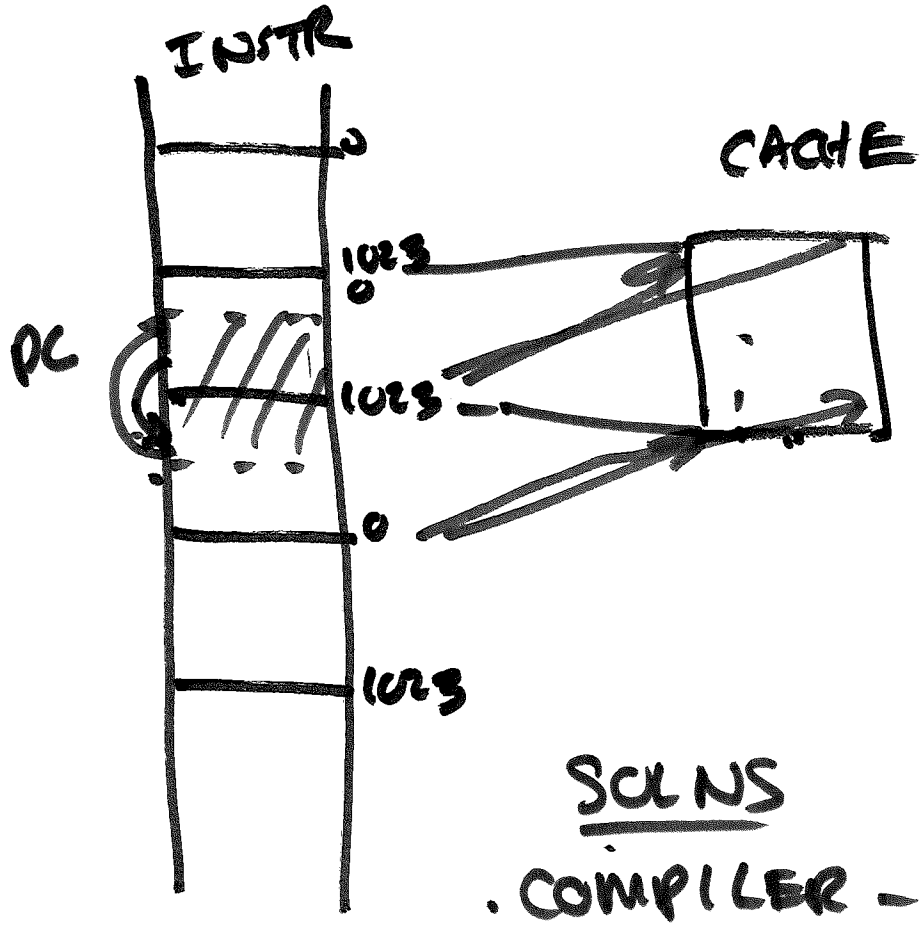
Figure C.41 The eight-stage pipeline structure of the R4000 uses pipelined instruction and data caches. The pipe stages are labeled and their detailed function is described in the text. The vertical dashed lines represent the stage boundaries as well as the location of pipeline latches. The instruction is actually available at the end of IS, but the tag check is done in RF, while the registers are fetched. Thus, we show the instruction memory as operating through RF. The TC stage is needed for data memory access, since we cannot write the data into the register until we know whether the cache access was a hit or not.

University of Idaho

WRITES -

1. WRITE THROUGH - ALWAYS WRITE TO MAIN MEM & CACHE
   - ALWAYS COHERENT
   - LOSE SOME CACHE BENEFIT, SIMPLE
2. WRITE BACK
   - WRITE ONLY TO CACHE
   - WRITE BLOCK TO MAIN MEM WHEN NECESSARY.
   - MORE CACHE - COMPLEX TO IMPLEMENT.

CAN USING CACHE MAKE CODE RUN SLOWER?



YES!

- FREQUENTLY CROSS CACHE BOUNDARIES
- MUCH SWAPPING IN & OUT OF MEMORY.
- THRASHING

SOLNS

- COMPILER - PUT CODE IN ONE CACHE BLOCK
- ORGANIZATION - "ASSOCIATIVE" FLEXIBLE BOUNDARIES