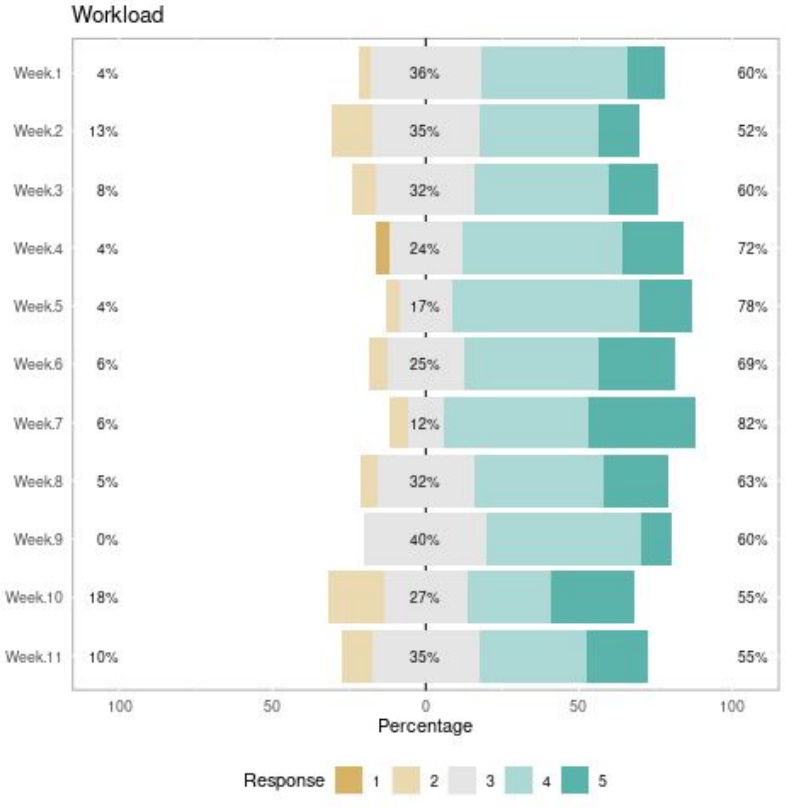
0x15 - Other Architectures

ENGR 3410: Computer Architecture

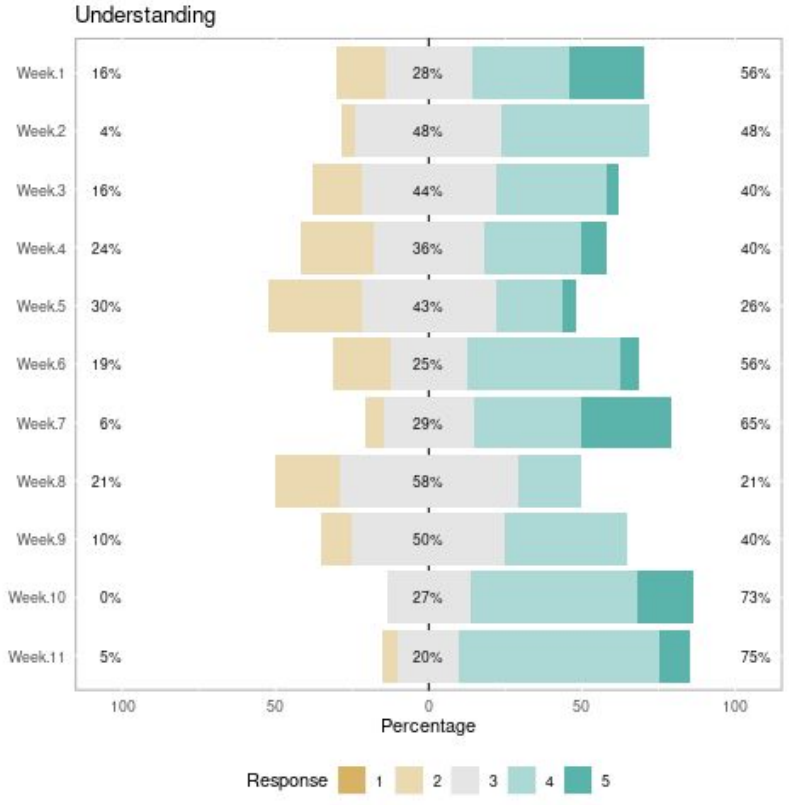
Jon Tse

Fall 2020

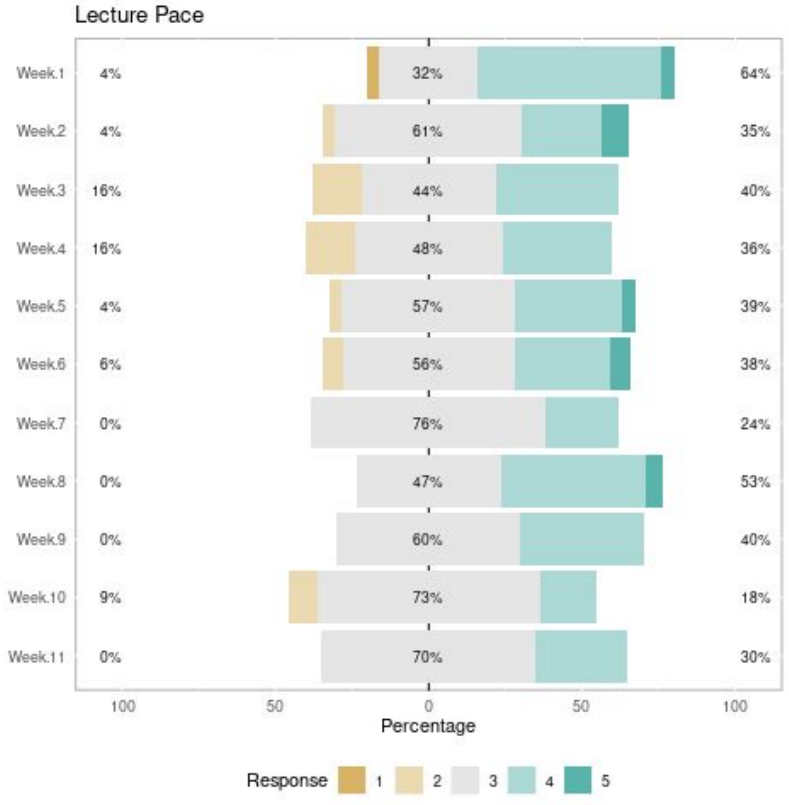
Feedback - Workload



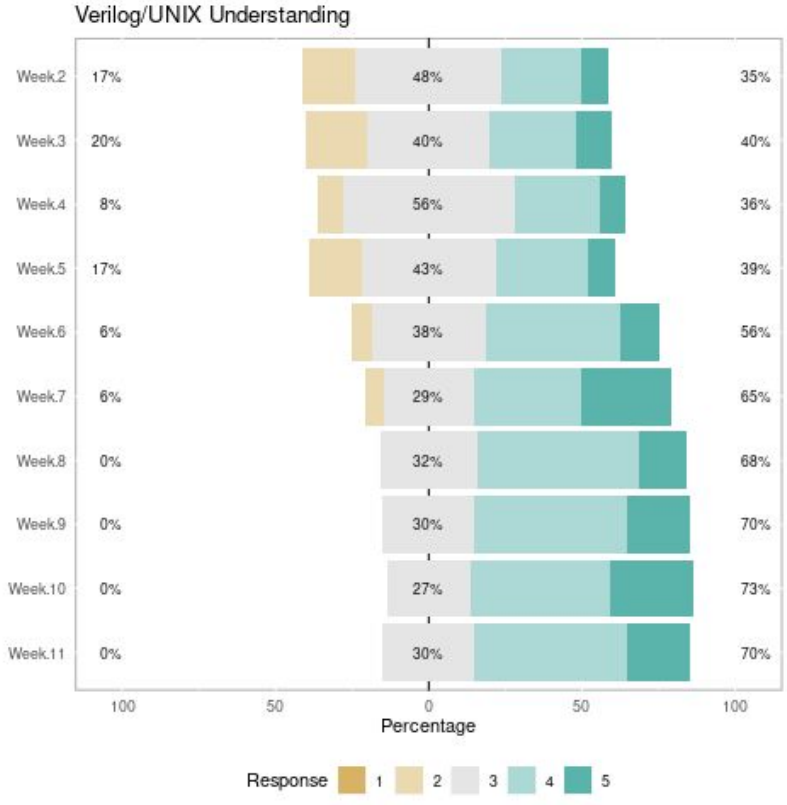
Feedback - Understanding



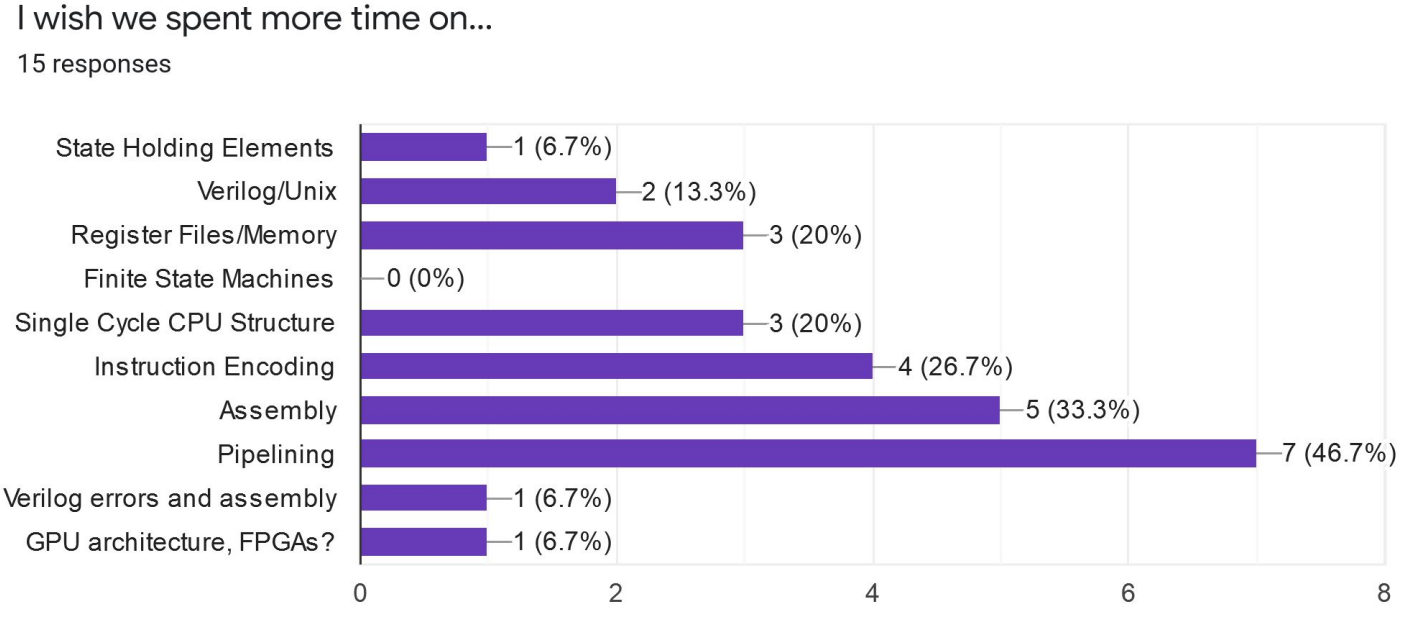
Feedback - Pace



Feedback - Tools



Feedback - More Time...

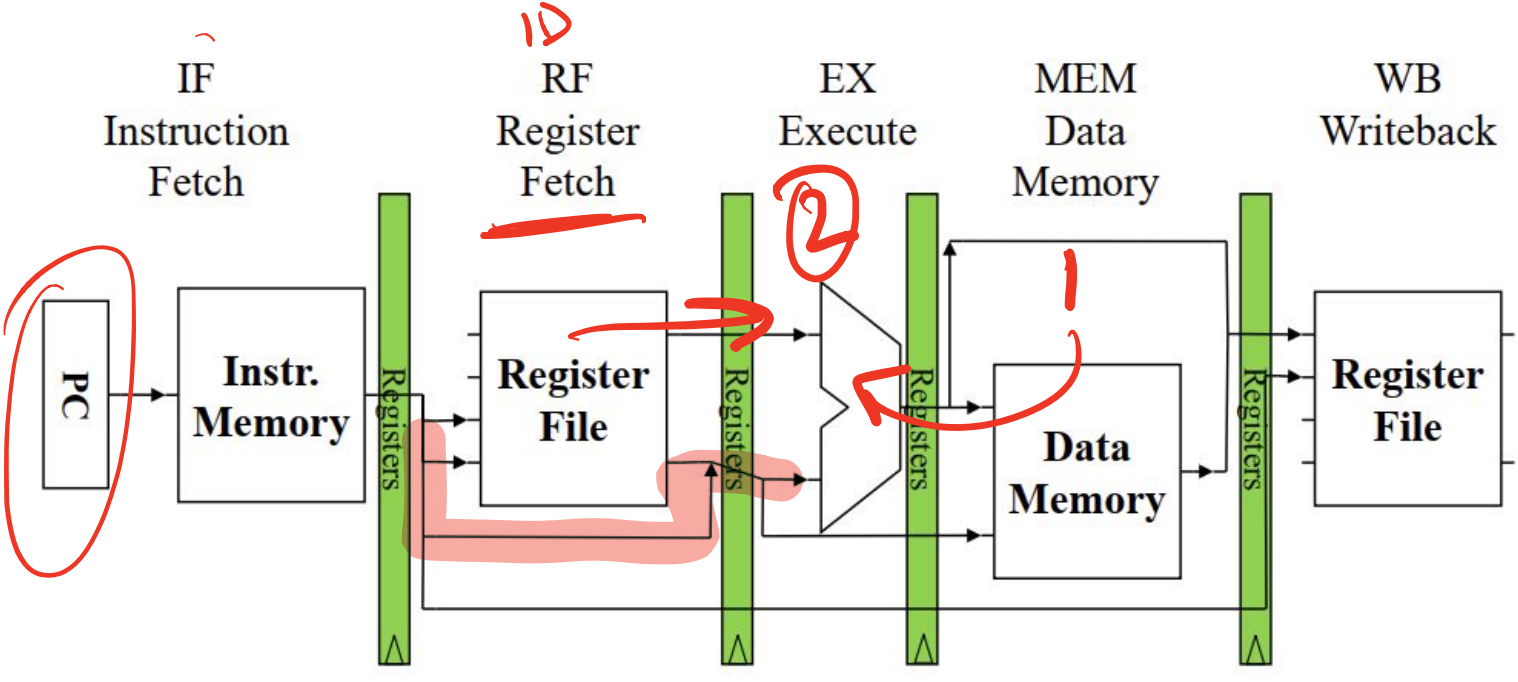


Feedback - Top of Mind

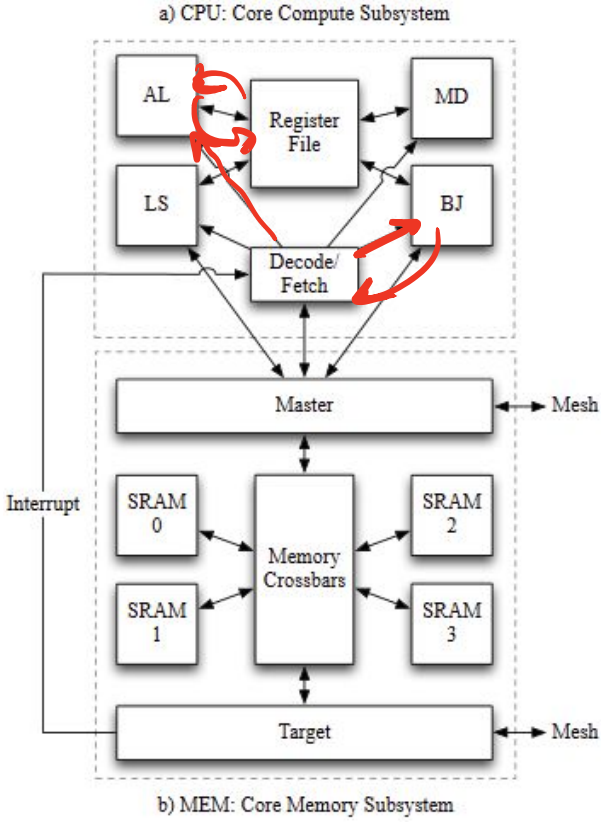
Housekeeping

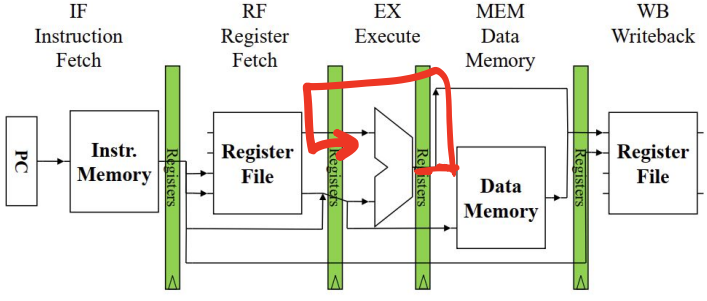
- Sign up for a Final Project Time Slot!
- Keep these slots for exam period?
- Reminder to petition NINJAs for advising
- Ping us early and often!

Our CPU Architecture



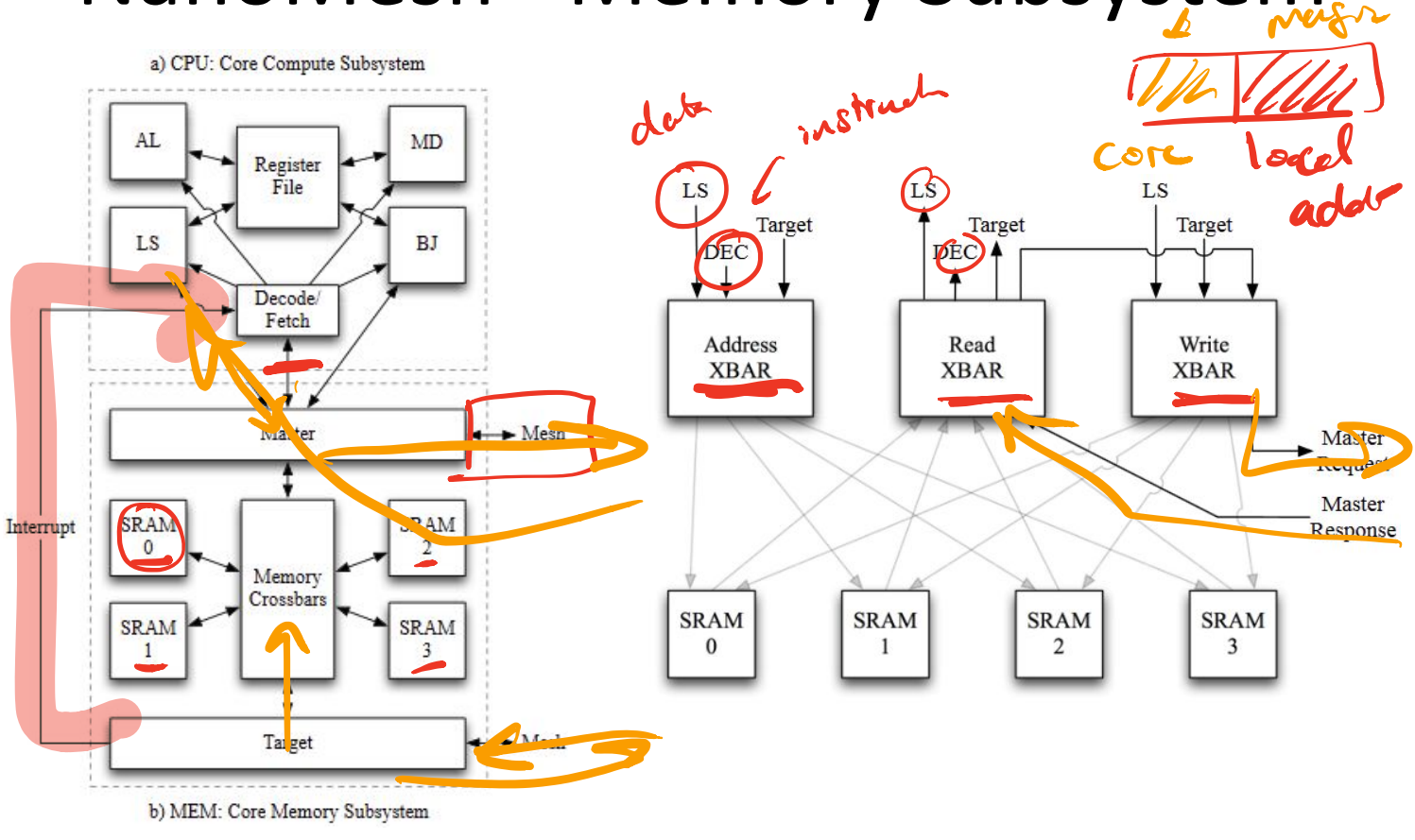
NanoMesh - Kilo Core System



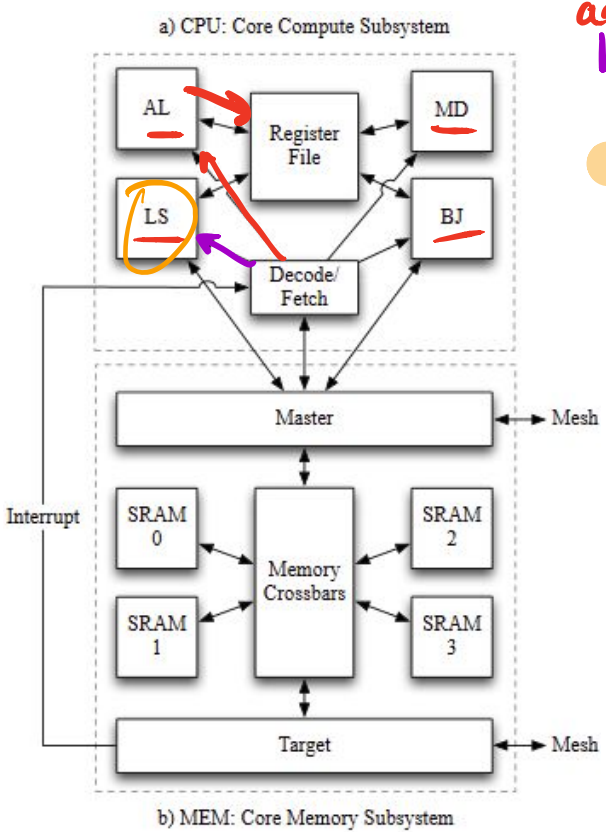


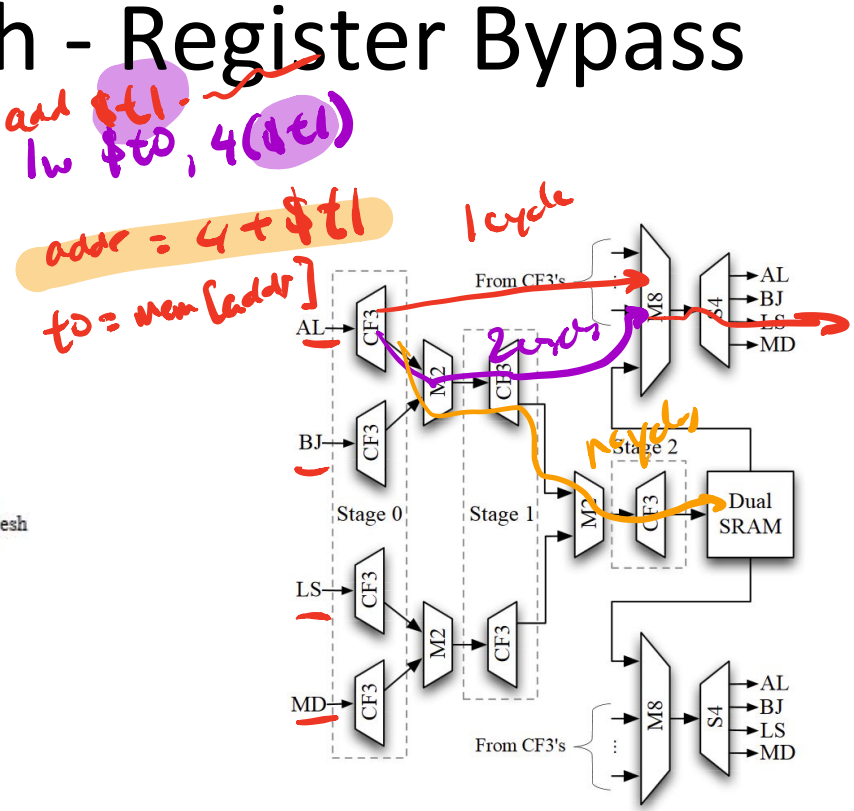
Unit	R3000	New
DEC	NOP RFE SYSCALL	HALT WAIT
AL	ADDU ADDIU SUBU AND ANDI OR ORI NOR XOR XORI LUI SLL SLLV SRA SRL SRLV SLT SLTI SLTU SLTIU	
MD	MULT MULTU DIV DIVU MFHI MFLO MTHI MTLO	
BJ	BEQ BGEZ BGTZ BLEZ BLTZ BNE BGEZAL BLTZAL J JAL JALR JR MFC0 MTC0	
LS	LB LBU LH LHU LW SB SH SW	CF4 CF16 CT4 CT16
Omitted	ADD ADDI SUB LWL LWR SWL SWR	

NanoMesh - Memory Subsystem

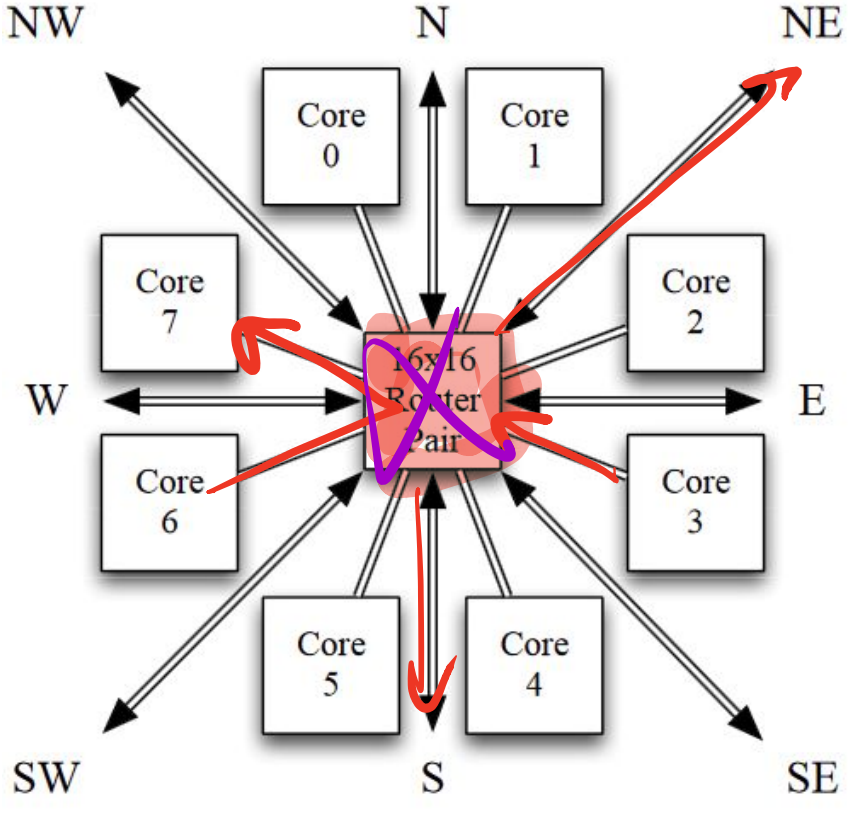


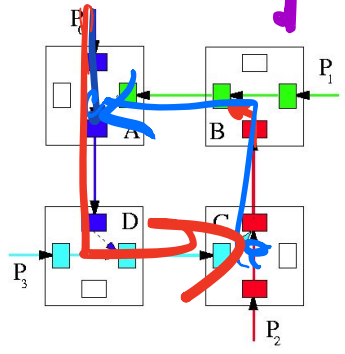
NanoMesh - Register Bypass

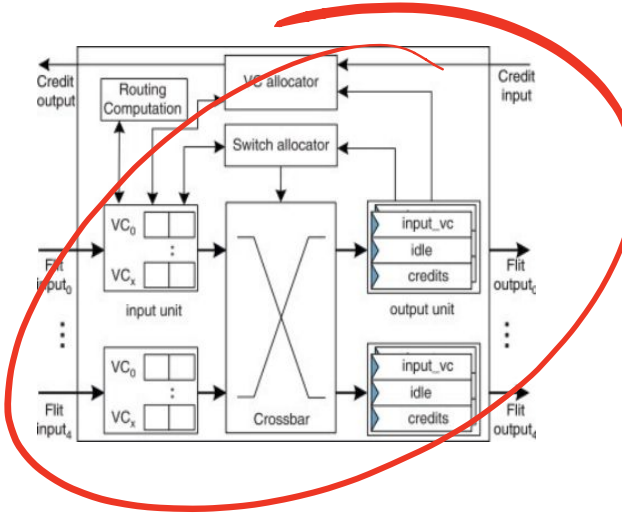




NanoMesh - Mesh

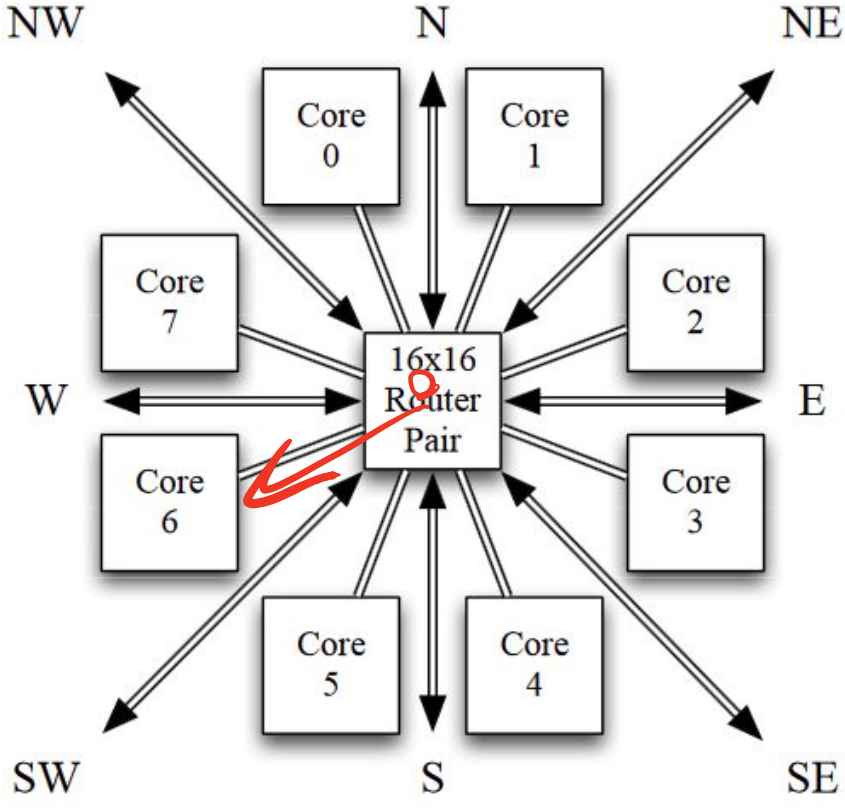


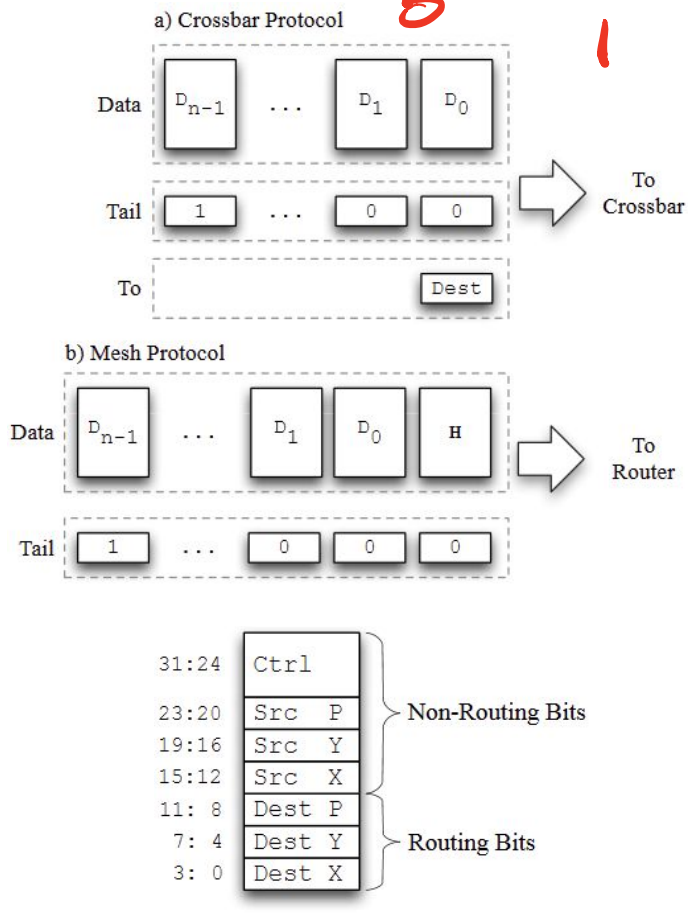




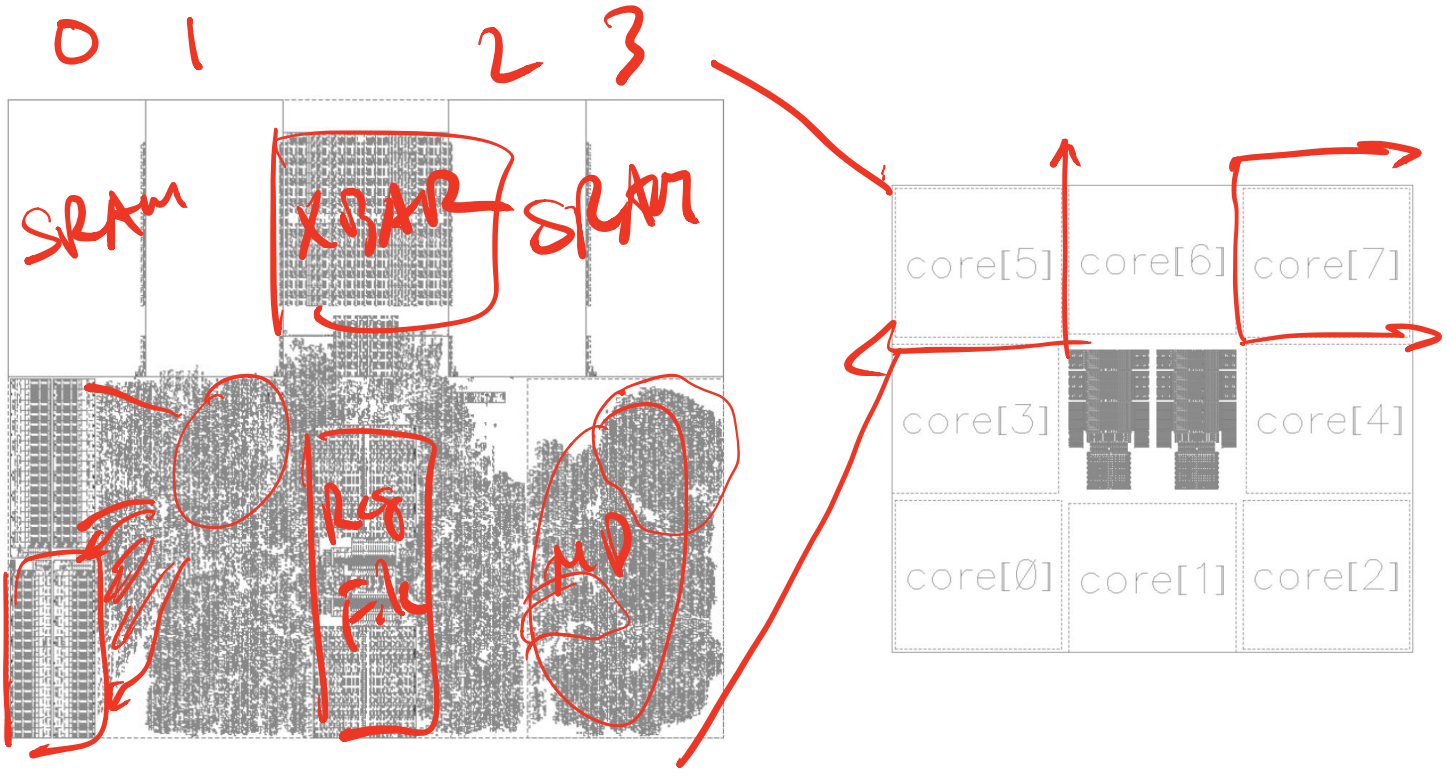
http://pages.cs.wisc.edu/~tvrdik/8/html/Section8.html
https://www.sciencedirect.com/topics/computer-science/output-virtual-channel

NanoMesh - Mesh





NanoMesh - Floorplan



NanoMIPS - New Instructions

- `WAIT` — Stalls fetch until dirty memory flag set
  - Prevents busy-waiting
  - Allows for rudimentary message-passing constructs
- `HALT` — Stalls instruction fetch until `Interrupt`
  - NanoMIPS cores bootstrap into `HALT` state
  - Exposes clock-gating like behavior directly to the programmer
- Memcopy Instructions
  - `CF4, CF16` — Copies 4 or 16 words from local memory to remote memory address.
  - `CT4, CT16` — Copies 4 or 16 words from remote memory to local memory.
  - Addresses must be block-size aligned.
  - Supports all addressing modes
  - Supports dirty memory flag setting

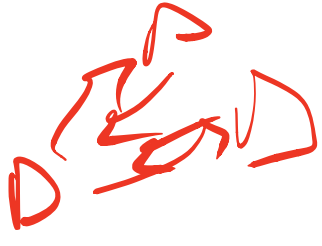
NanoMesh - Example Program

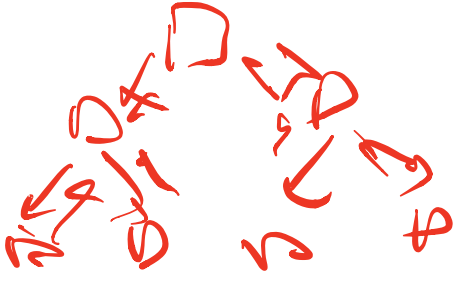
Core A — Control

1. Write Program to Core B
2. Fire `Interrupt` on Core B
3. Send data to Core B
4. `WAIT`
5. Continue execution

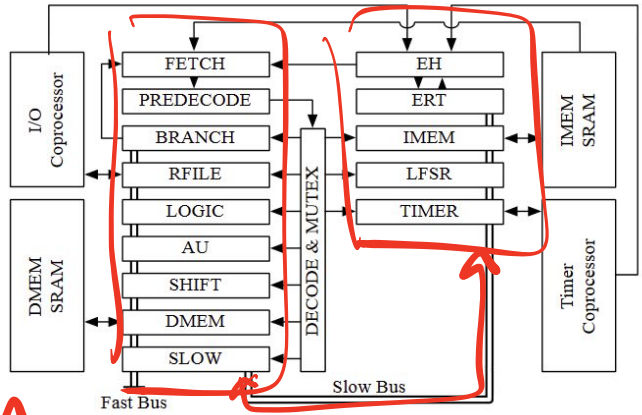
Core B — Worker

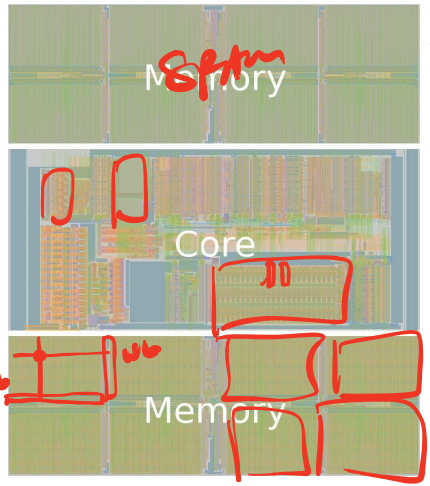
1. `HALT`
2. Initialize
3. `WAIT`
4. Do work on data
5. Send result to Core A
6. `HALT`

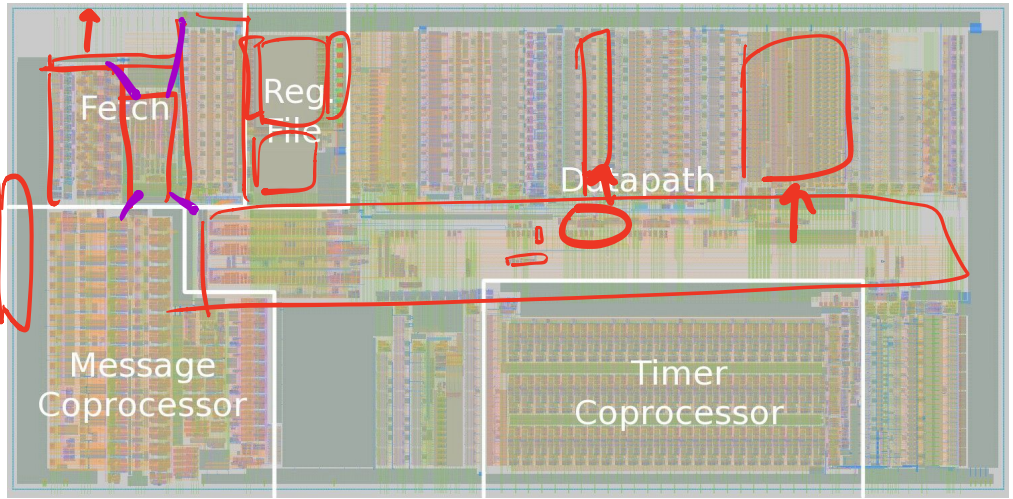




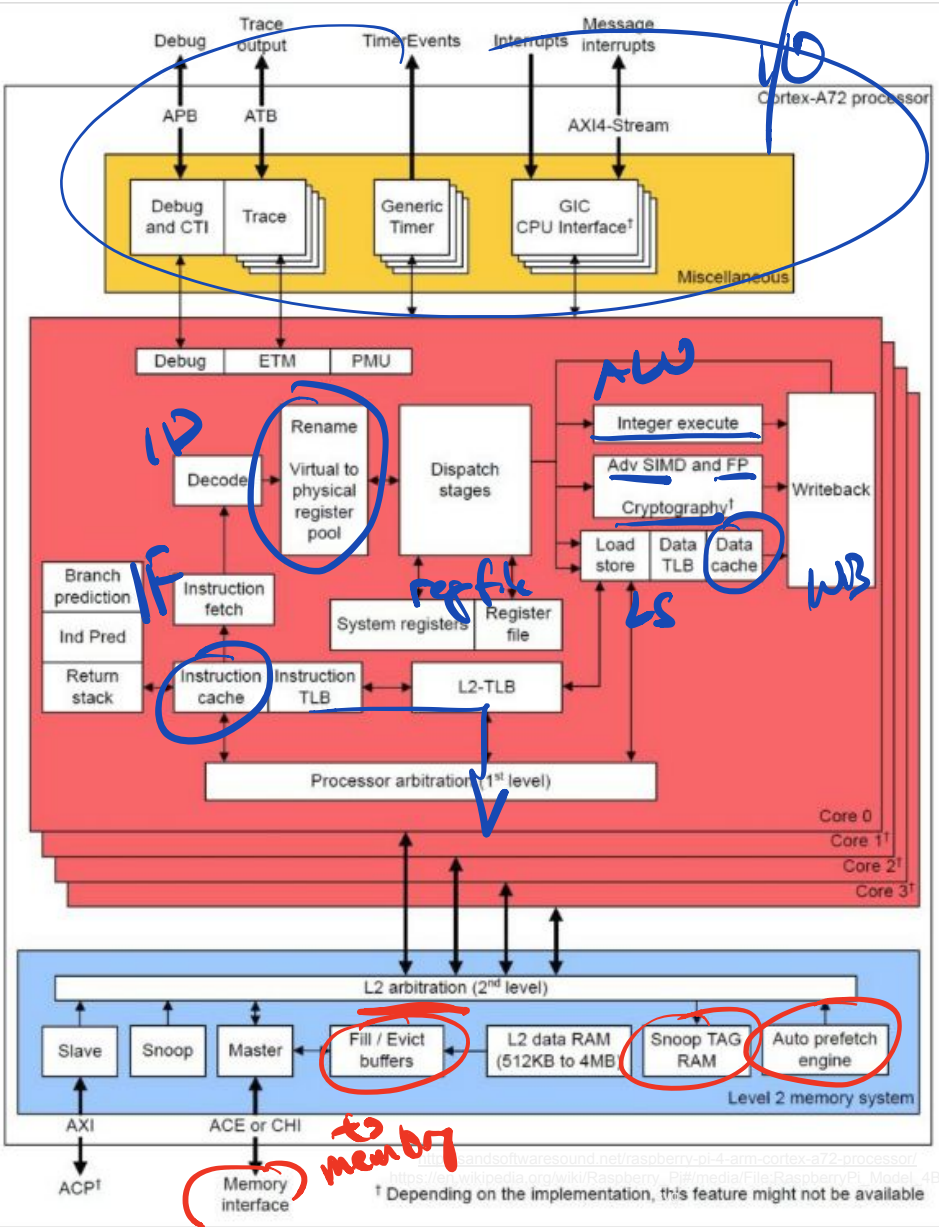
ULSNAP - Low Power CPU

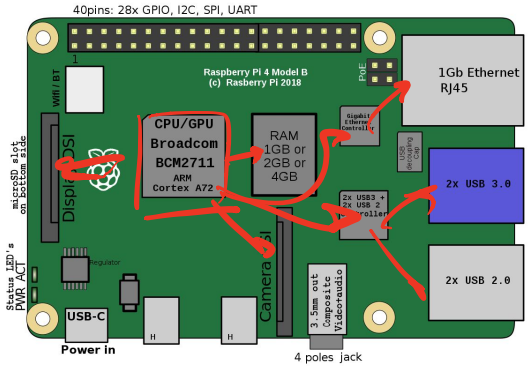




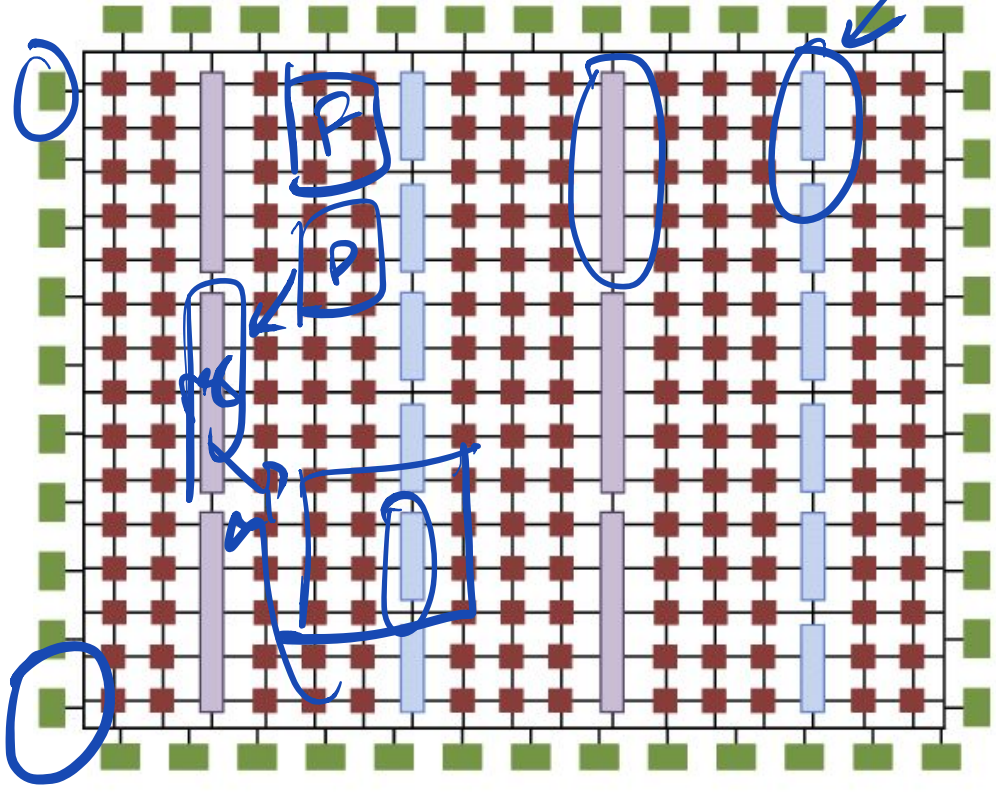


ARM Cortex A72

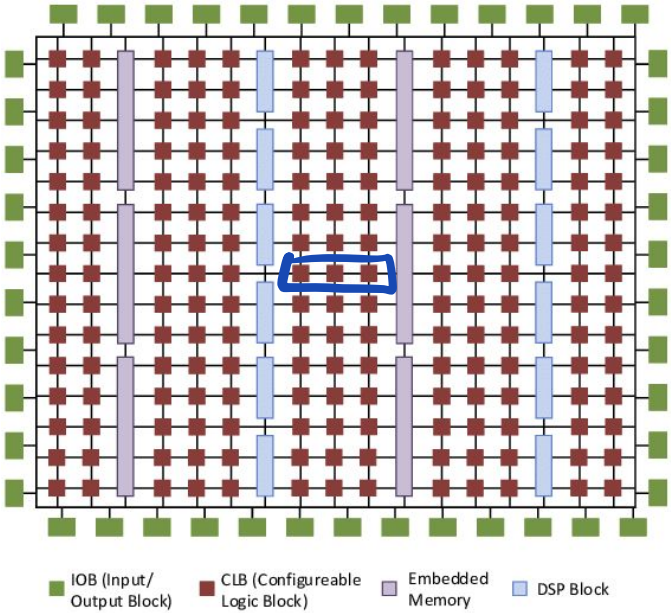


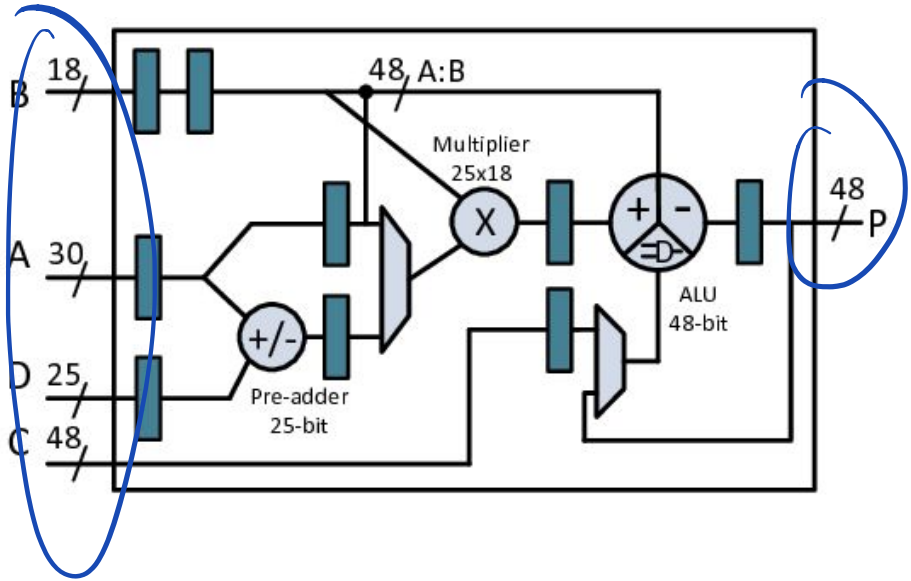


FPGA Architecture

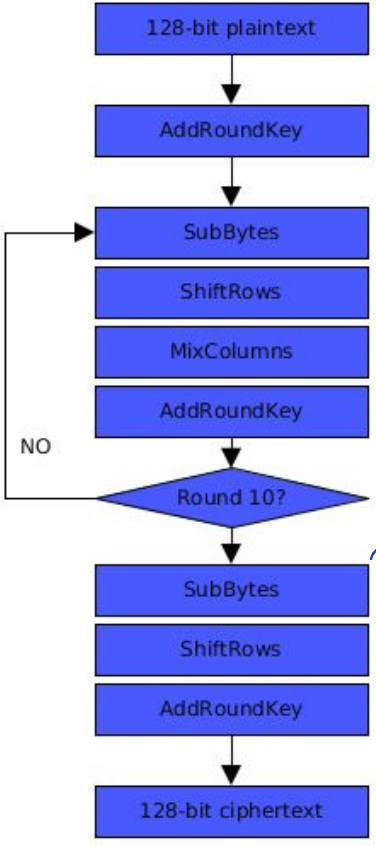


FPGA DSP Block



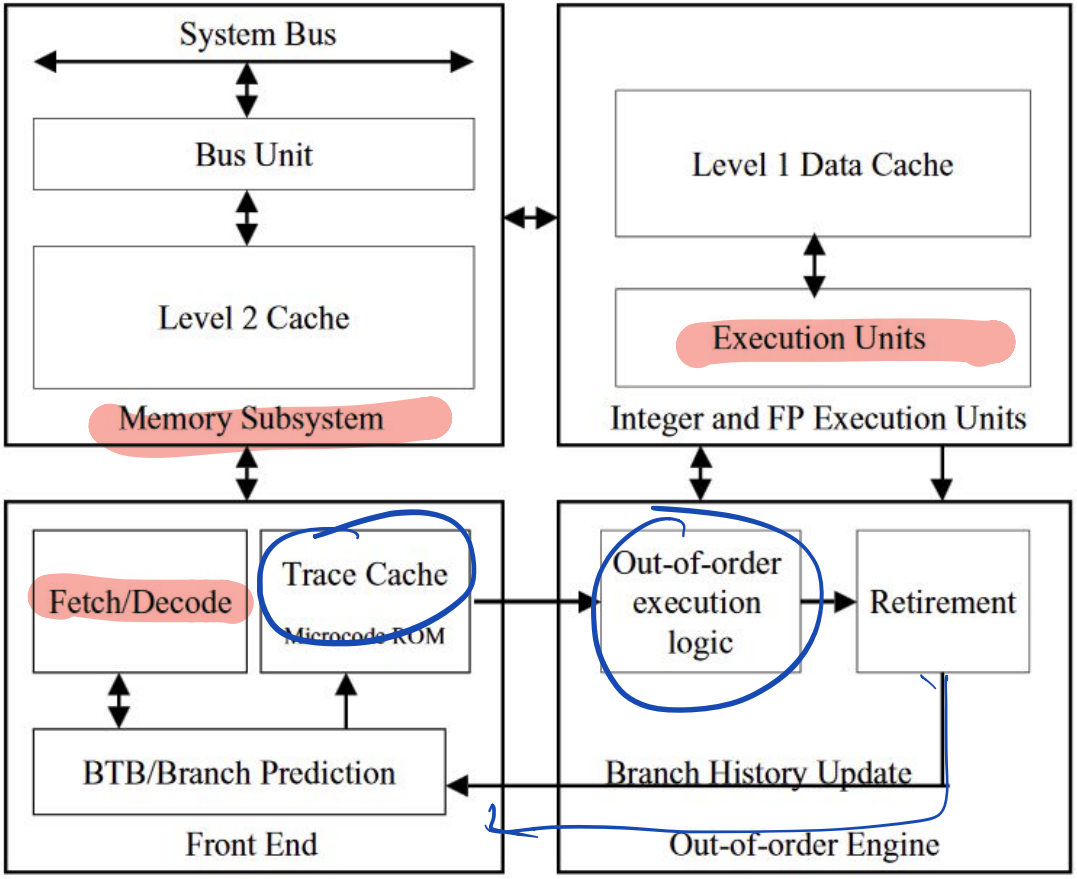


AES on FPGA




```
; ***************************
; ShiftRows and SubBytes
; ***************************
; F(16)((B*)x)[(i%4)+(((i/4)-(i%4))%4)*4]=S(((B*)s)[i]);
    pushad
    mov     cl, 16
shift_rows:
    lodsb                       ; al = S(s[i])
    call    sub_byte
    push    edx
    mov     ebx, edx            ; ebx = i%4
    and     ebx, 3              ;
    shr     edx, 2              ; (i/4 - ebx) % 4
    sub     edx, ebx            ;
    and     edx, 3              ;
    lea     ebx, [ebx+edx*4]    ; ebx = (ebx+edx*4)
    mov     [edi+ebx], al       ; x[ebx] = al
    pop     edx
    inc     edx
    loop    shift_rows
    popad
```

Pentium 4



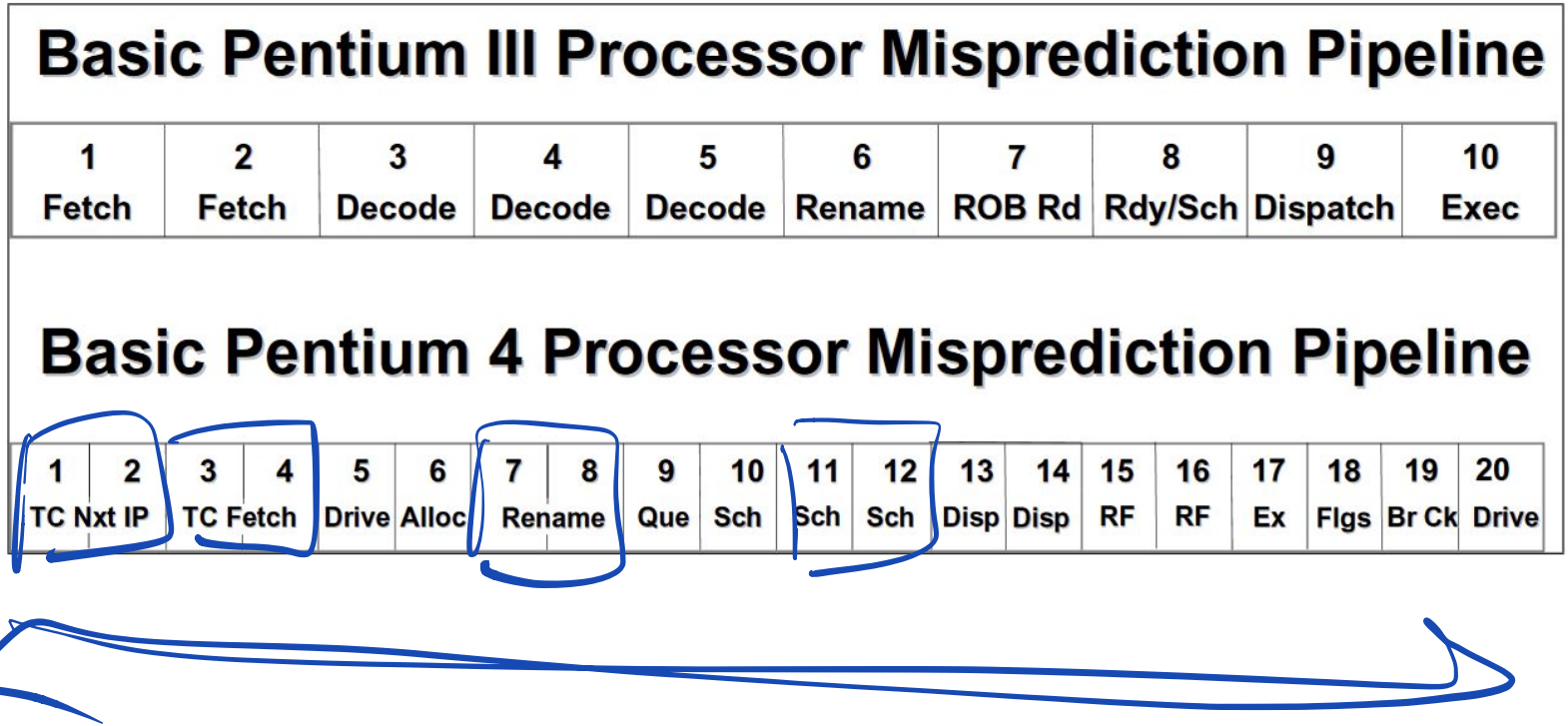
http://www.ecs.umass.edu/ece/koren/ece568/papers/Pentium4.pdf

Pentium 4

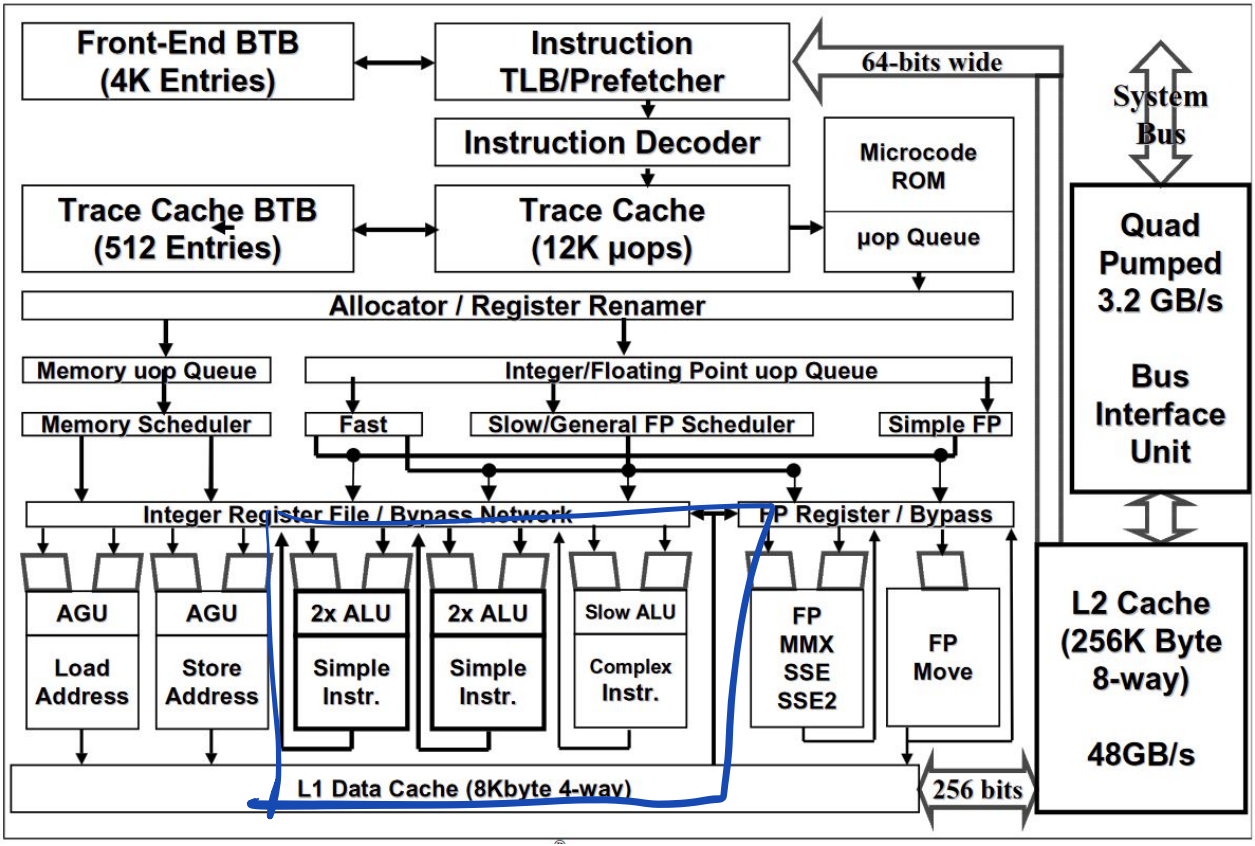
Basic Pentium III Processor Misprediction Pipeline

1	2	3	4	5	6	7	8	9	10
Fetch	Fetch	Decode	Decode	Decode	Rename	ROB Rd	Rdy/Sch	Dispatch	Exec

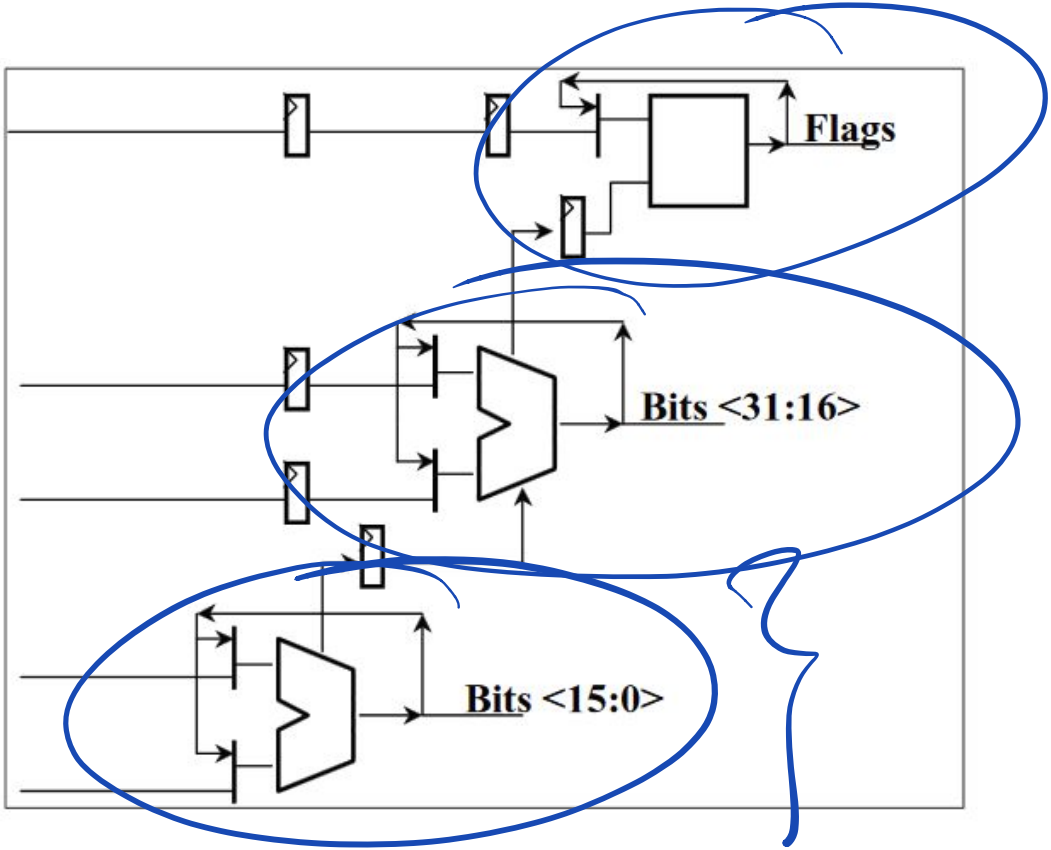
Pentium 4



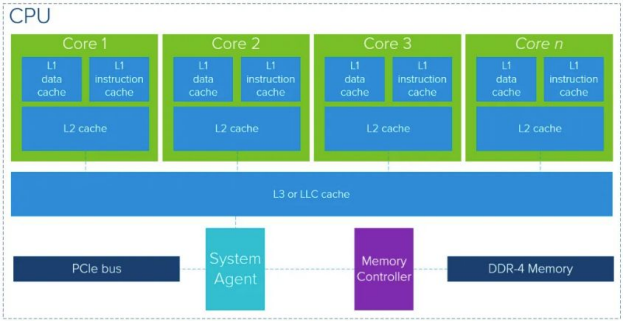
Pentium 4

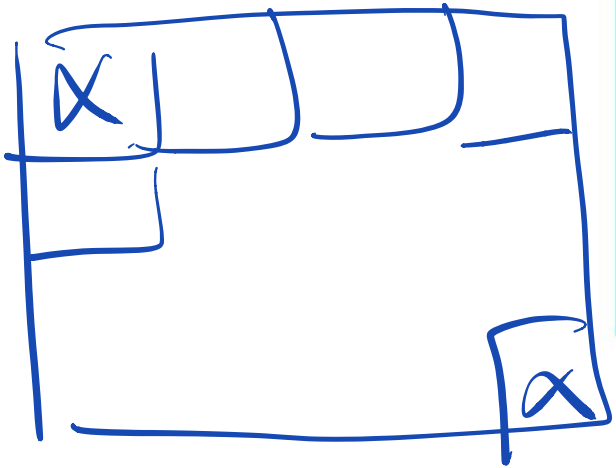


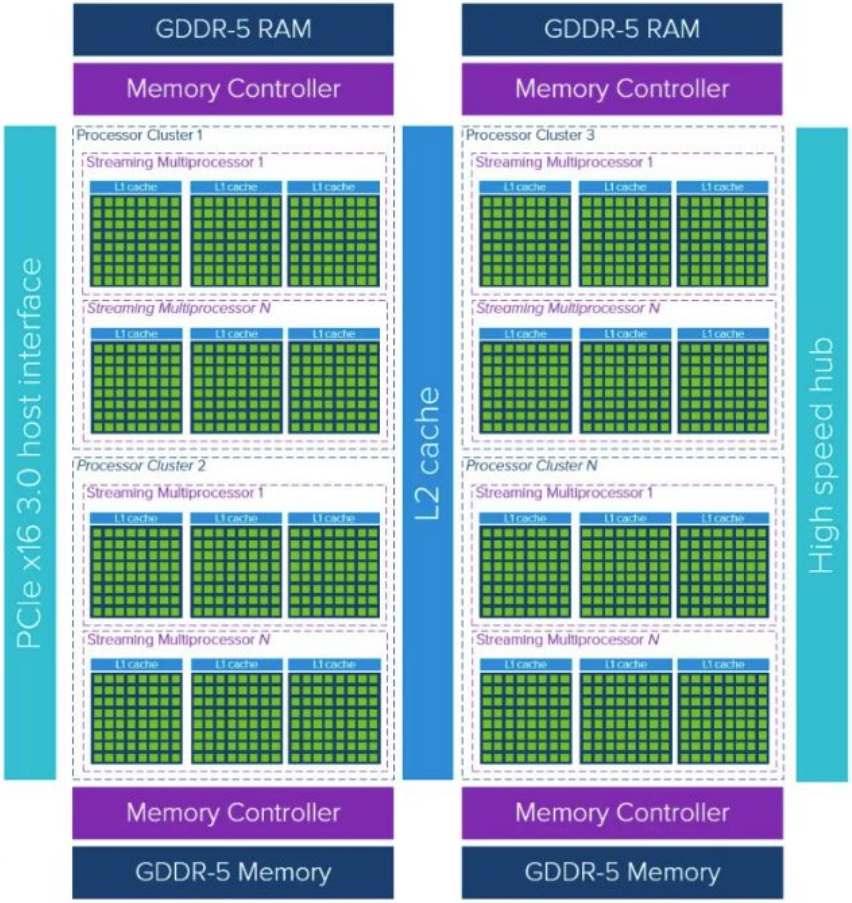
Pentium 4



GPUs







https://nielshagoort.com/2019/03/12/exploring-the-gpu-architecture/