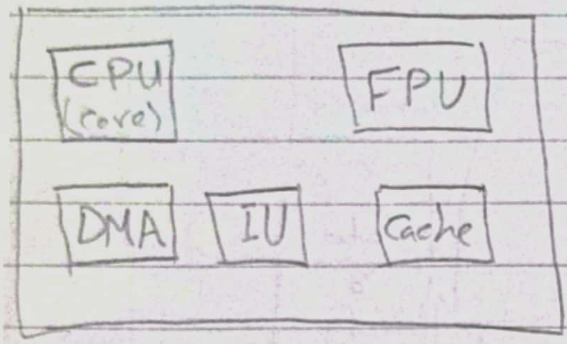
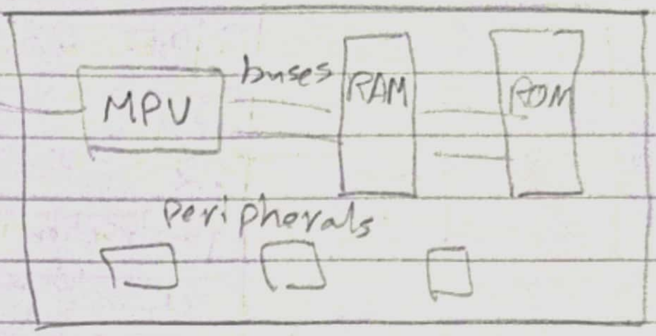
MPU



MCU



- FPU ⇒ Floating point unit — هي اللي بتعمل عمليات الـ Float numbers
طب لو مش موجودة؟ الـ CPU هو اللي هينفذها بس هياخد وقت ومش هينفذها بدقة

- DMA ⇒ Direct Memory access — هو بينقل داتا مباشرة
مش موجودة في كل الـ MCUs فاللي بينقل داتا طبيعي هو الـ CPU بس نقل الداتا مفيهوش أي process عليها غير النقل بس فهنا تيجي وظيفة الـ DMA وهو إنه بينقل الداتا بدل الـ CPU ويوفر وقت

أكبر size →

- Cache Memory — register cache RAM

← أسرع

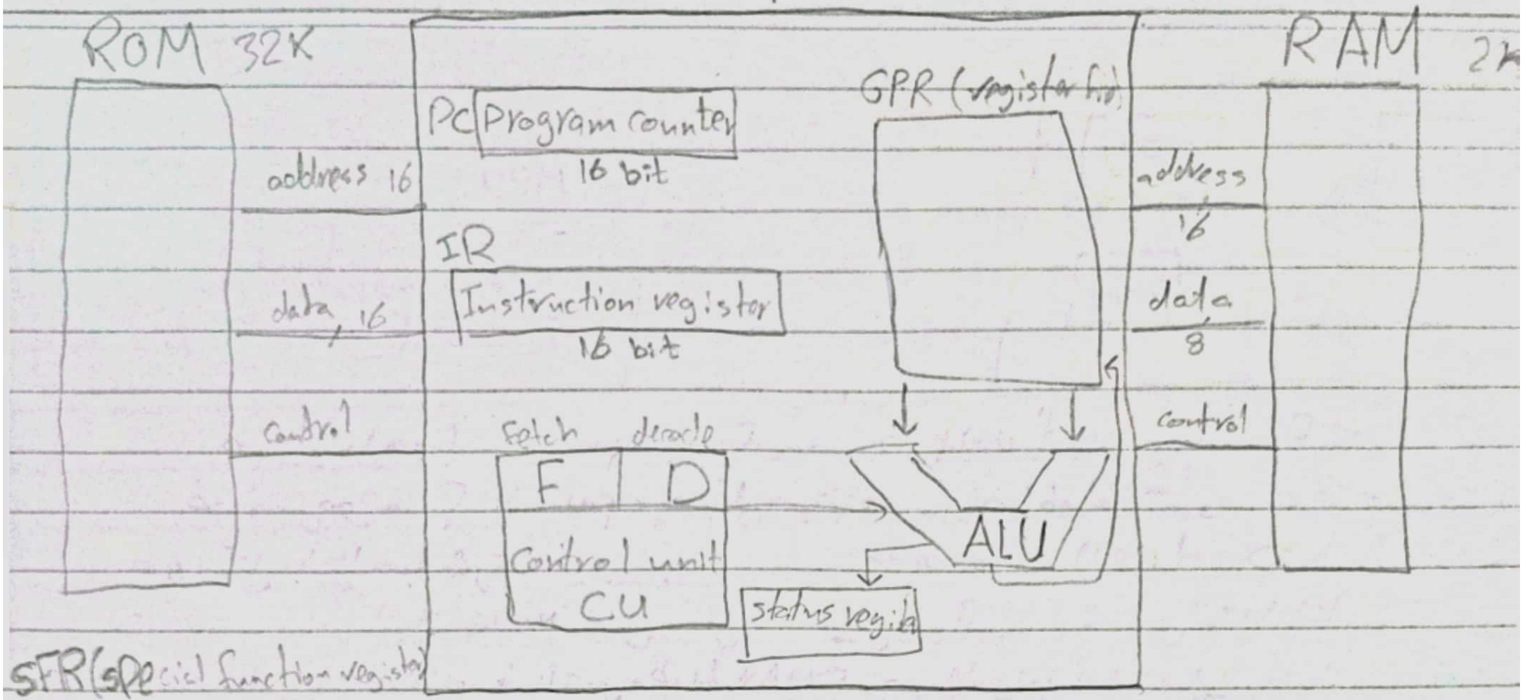
- حاجة زي إيه؟ عندي مكتبة مثلاً فيها كتب كتير وعايز أذاكر C فجبت كتب الـ C جمبي الـ... منهم كتاب فتحته وذاكرت فيه:
- الكتاب اللي فتحته ← register
- الكتب اللي جمبي ← cache
- الكتب اللي في المكتبة ← RAM

- عندي مصطلحين cache hit , cache loss
البروسيسور بيبقى حاطط حاجات محتاجها موجودة في الـ cache فلو أنا بشتغل مثلاً على داتا x = 5;
فلو الـ x موجودة في الـ cache فكده cache hit , ولو مش موجودة cache loss فلو مثلاً من وسط 100 سطر لقيت 20 hit و 80 loss فكده الـ algorithm اللي شغال بيها مش كويسة

- الـ microcontroller مش بيبقى فيه cache فأعوضها إزاي؟ لأن الـ RAM بتبقى قريبة منه كده
- أكتر الـ registers
- الـ RAM تبقى من نوع أسرع زي SRAM

CPU

AVR



- Program Counter: holds the address of the next instruction
- Instruction register: Contains the instruction after getting it from memory

Fetch circuit

- فلنفرض مثلاً إن في البداية PC شايل 0 address الي هو أول حاجة هتتنفذ. أول stage عندنا هي ال Fetch. تقوم دايرة ال Fetch تاخد ال address الموجود في ال PC تحطه على ال address bus وترمي على ال Control bus تقول إنها عايزة تعمل read الداتا اللي في ال address ده، بعدها ال ROM تقوم حاطة ال instruction اللي في ال address ده على ال data bus وينتقل على ال IR، بعدها ال address اللي موجود في ال PC يزيد واحد ويروح على ال instruction اللي بعدها والدايرة تكمل.
increment ← يا 4 يا 2 بايت

- ال decoding unit بتفهم/تترجم ال instruction اللي جه في ال IR

opcode ← ld R_1, X → operands

هتفهم إن هي load instruction

Execute

- بعدها ال decoding unit ترجمت، ال CU بتفهم إن فيه عملية load من address X مثلاً من الرام مثلاً، فتقوم حاطة ال address ده على ال address bus وترمي على ال control bus تقول إنها عايزة تعمل read داتا، فبعدها الداتا هتنتقل على ال data bus وتروح من ال register file تتخزن في R_1 register

- ALU: Arithmetic logic unit.
ليه ساكنين جنب ال ALU؟ عشان ال two operands يتبعتوا في نفس الوقت

size of register = RAM databus = ALU تقدر تعمل عمليات على قد إيه

في حالة ال avr ⇐ 8-bit

- Status register: Contains information about the result of the most recently executed arithmetic instruction.

Zero, Carry ... Flags ...

- Stack pointer: ... stack ...

⇒ ... the Function call ...

* Pipelining: Parallel instruction fetches and executions enabled by the Harvard architecture (...)

	Cycle 1	Cycle 2	Cycle 3	cycle 4	Cycle 5
Ins 1	F	D	E		
Ins 2		F	D	E	
Ins 3			F	D	E

⇐ ... Harvard ... Von-Neumann

Harvard	Von-Neumann
- Bus connects CPU to RAM and another Bus connects CPU to ROM - The CPU can read an instruction and perform a data memory access at the same time which speeds up execution time. - More hardware, cost, complexity	- There is a single data bus. - Each time the CPU fetches a program instruction, it may have to perform one or more read/write operations to data memory space, it must wait until these operations are completed before it can fetch and decode the next instruction - Simplicity and economy

* The AVR is a harvard arch. CPU which means it separates instruction memory and data memory. GCC was originally designed to support Von-Neumann arch which define a single storage structure to hold both instructions and data.

- compiler ... address ... segments ...

.text at 0x0
.data at 0x800000

number of instructions

RISC	CISC
Reduced Instruction set computers	Complex Instruction set computers
Limited number of instructions	Large number of instructions
Simple instructions	Complex instructions
Limited Addressing modes	Many addressing modes
Single cycle instruction execution	several cycles for instruction executi
More usage of RAM	Less usage of RAM
Large number of Registers For operations	Small number of Registers for operation Single register set
Can't operate directly on memory does not require external memory	operates directly on memory require external memory
Complex software	Simple software
Based on Harvard architecture	based on Von Neumann architecture
used in microcontrollers - Embedded syste ex. AVR, ARM, PIC	used in computer processors ex: AMD, intel
execution time is less	execution time is very high
small amount of Transistors (Yo)	Big amount of Transistors