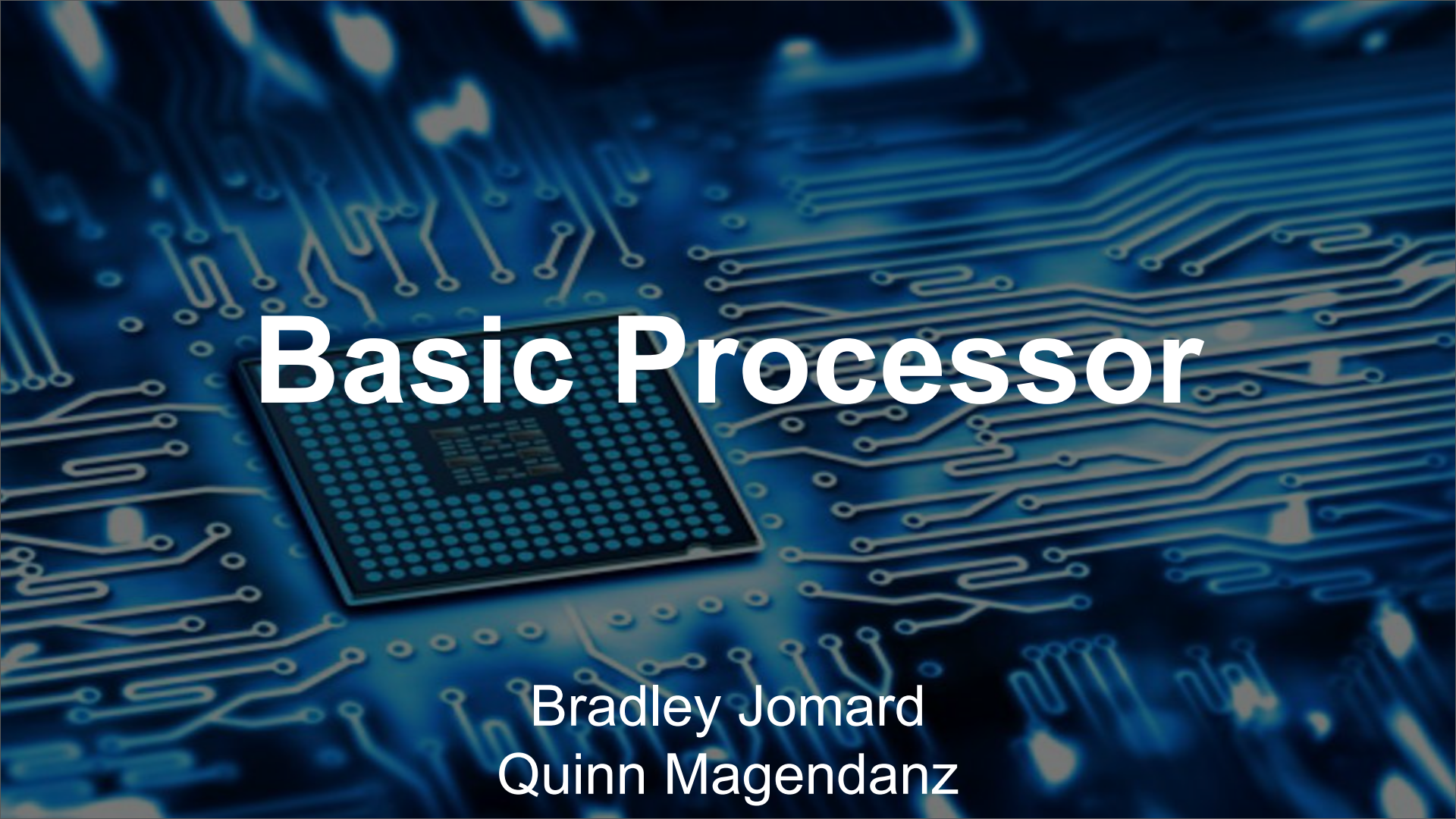
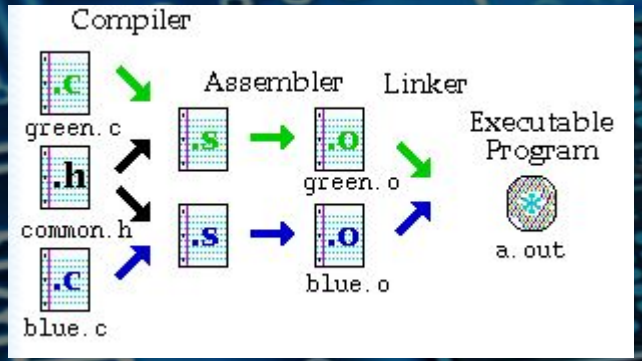


Basic Processor

Bradley Jomard
Quinn Magendanz

- How a processor works to run a simple program

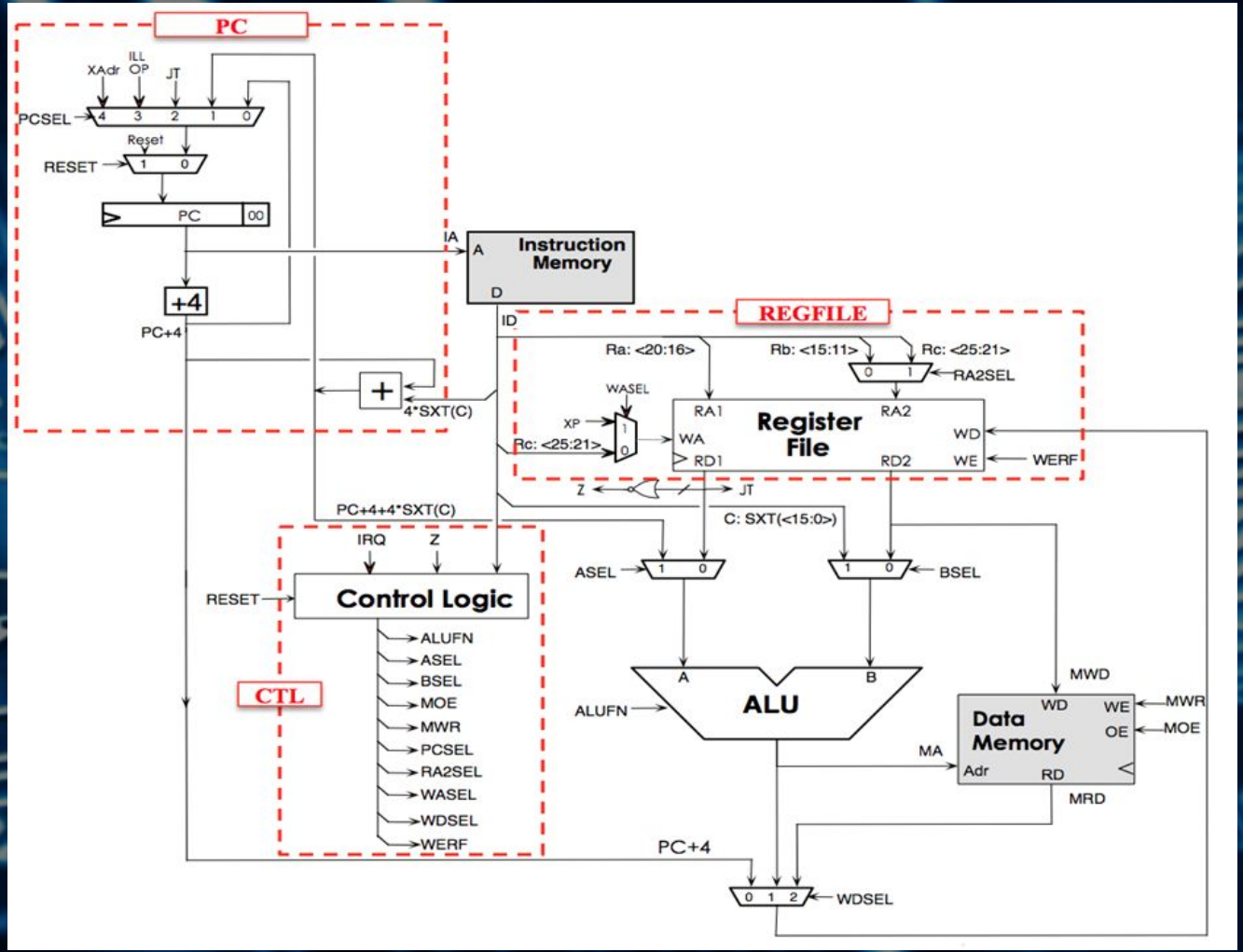


- Modern x86 Assembly Code

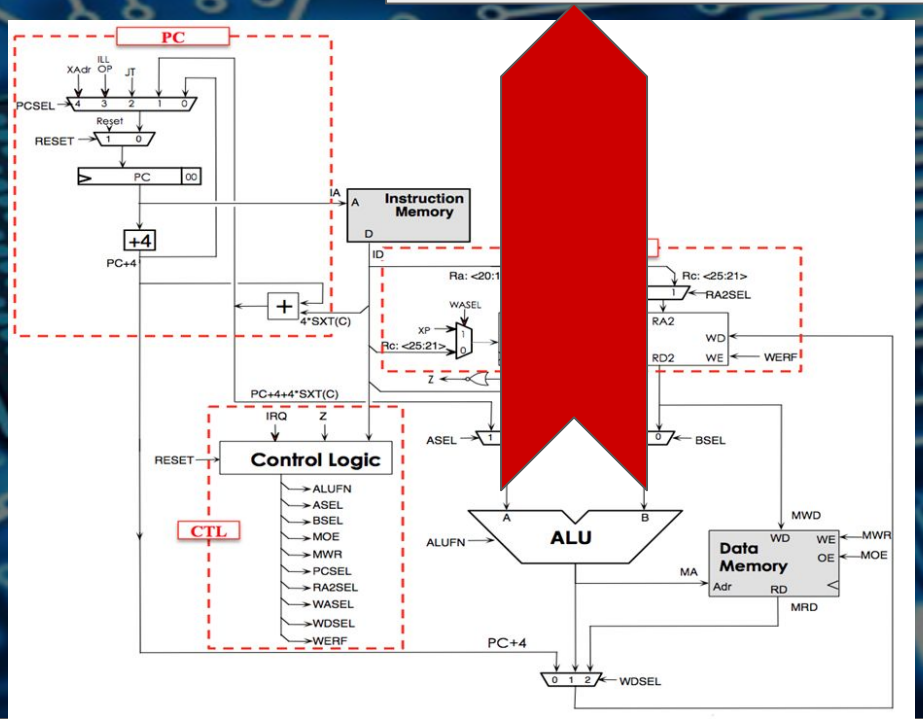
```
101  ;{341}:if(tmp >= array[_M])
102       mov EDX,dword ptr [EBP-000Ch]
103       mov EAX,dword ptr [EBP-0008h]
104       push     EAX
105       push     EDX
106       mov EBX,dword ptr [EBP+0008h]
107       mov EAX,EBX
108       push     EAX
109       mov EAX,dword ptr [EBP-0018h]
110       pop ESI
111       mov EDX,dword ptr [ESI+EAX*0008h]
112       lea EAX,dword ptr [EAX*0008h+4] ;low part
113       mov EAX,dword ptr [ESI+EAX]
114       push     EAX
115       push     EDX
116       call      FloatingPoint_Compare
117       cmp EAX,000000000h
118       neg EAX
119       setge    AL
120       test     AL,AL
121       jz  OFFSET $L000921
122
123  ;{342}:_F= _M + 1;
124       mov EAX,dword ptr [EBP-0018h]
125       push     EAX
126       mov EAX,000000001h
127       pop EDX
128       add EAX,EDX
129       mov dword ptr [EBP-0010h],EAX
130       jmp OFFSET $L000922
```

- Reduced Instruction Set

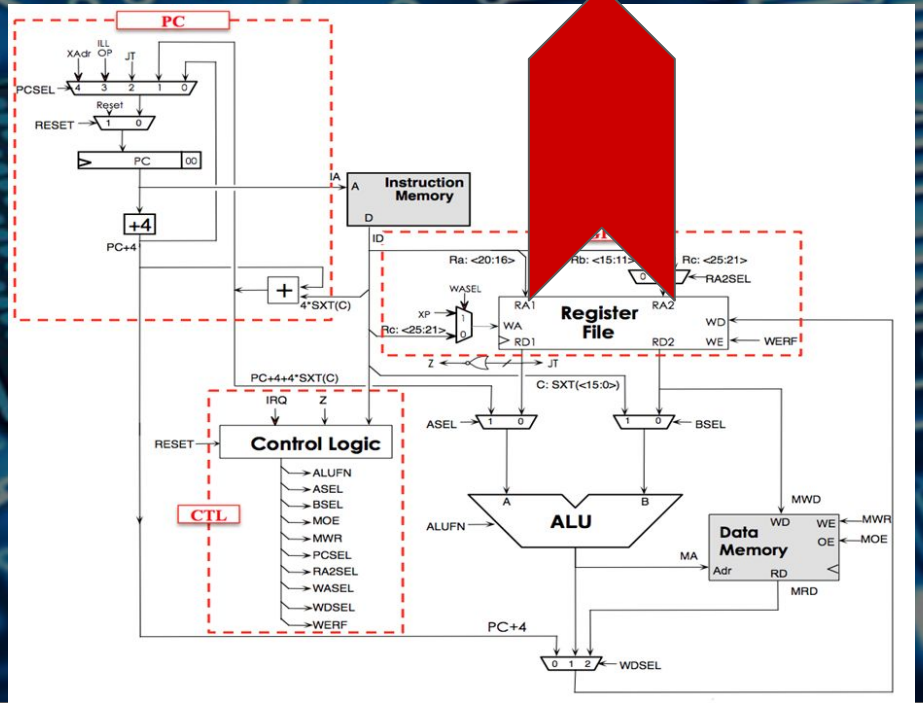
LD	ST
ADD	SUB
AND	OR
ADDC	SUBC
ANDC	ORC
LDR	JMP
MUL*	DIV*
XOR	XNOR
MULC*	DIVC*
XORC	XNORC
BEQ	BNE
CMPEQ	CMPLT
SHL	SHR
CMPEQC	CMPLTC
SHLC	SHRC


PC
XAdr
ILL OP
JT
PCSEL
4 3 2 1 0
Reset
RESET
1 0
PC
00
+4
PC+4
IA
A
Instruction Memory
D
ID
+
4*SXT(C)
REGFILE
Ra: <20:16>
Rb: <15:11>
Rc: <25:21>
0 1
RA2SEL
WASEL
XP
Rc: <25:21>
1
0
RA1
WA
RD1
Register File
RA2
RD2
WD
WE
WERF
Z
JT
PC+4+4*SXT(C)
C: SXT(<15:0>)
IRQ
Z
RESET
Control Logic
ALUFN
ASEL
BSEL
MOE
MWR
PCSEL
RA2SEL
WASEL
WDSEL
WERF
CTL
ASEL
1 0
1 0
BSEL
A
B
ALUFN
ALU
MWD
WD
WE
MWR
Data Memory
OE
MOE
MA
Adr
RD
MRD
PC+4
0 1 2
WDSEL

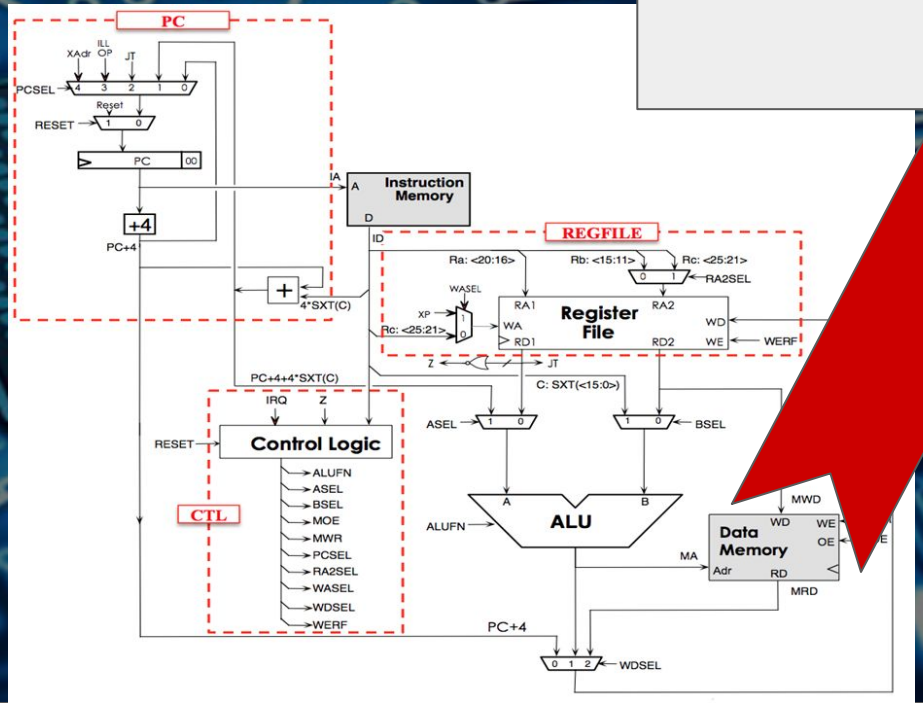
Arithmetic Logic Unit :
Performs basic logic on two register values



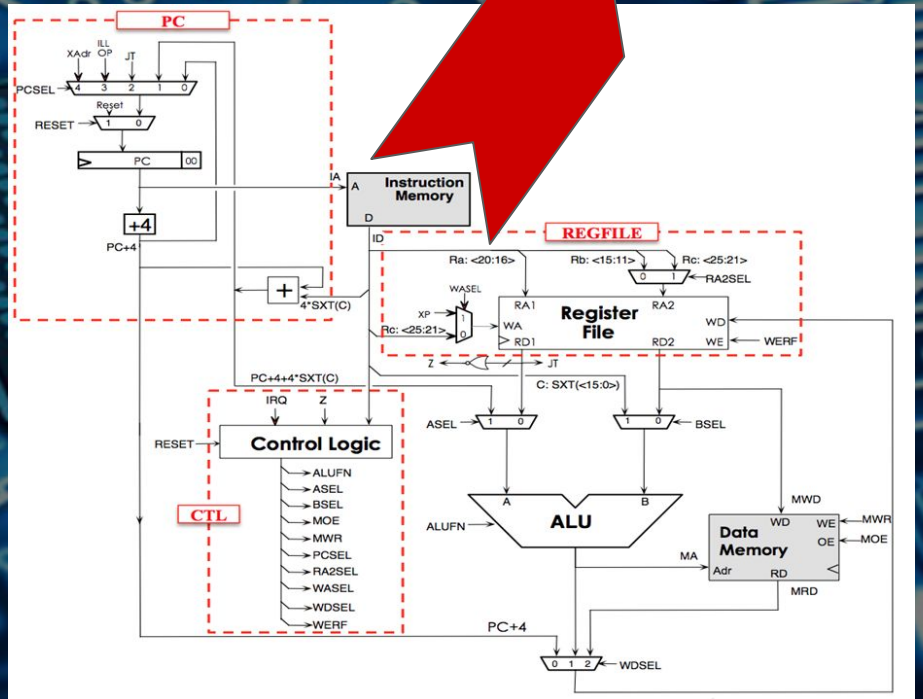
Contains thirty two 32-bit registers



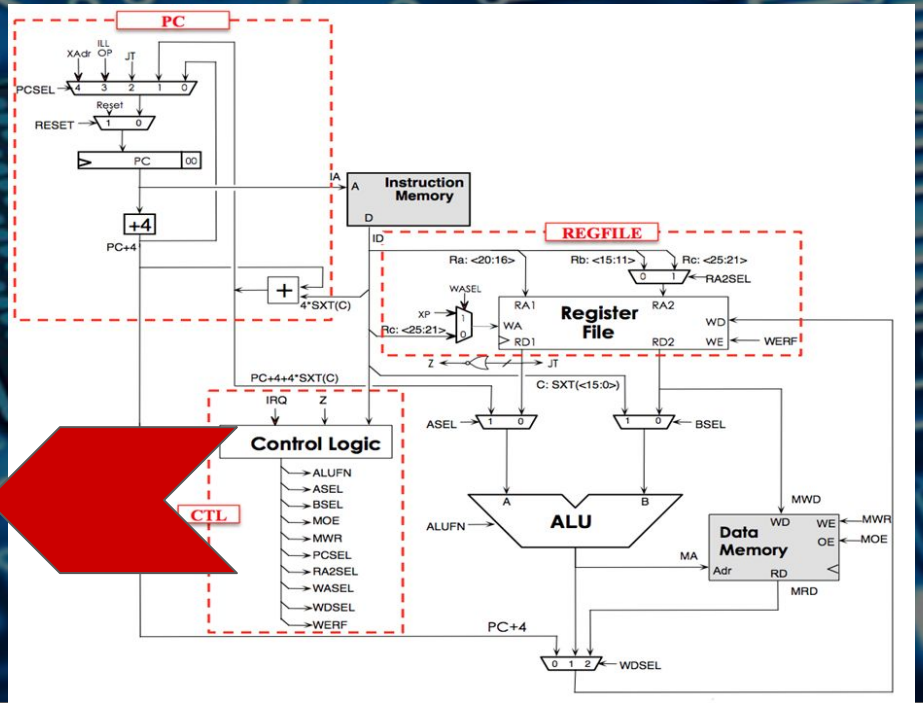
Stores values in RAM by address



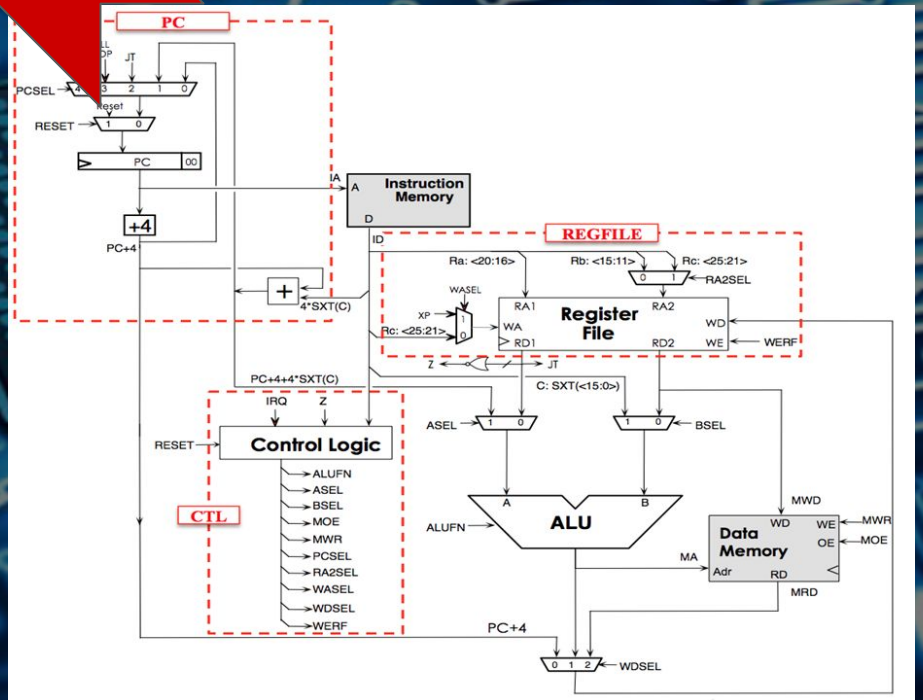
Section of RAM which stores the linear set of instructions that make up a program



Stores the processor state corresponding to each instruction value

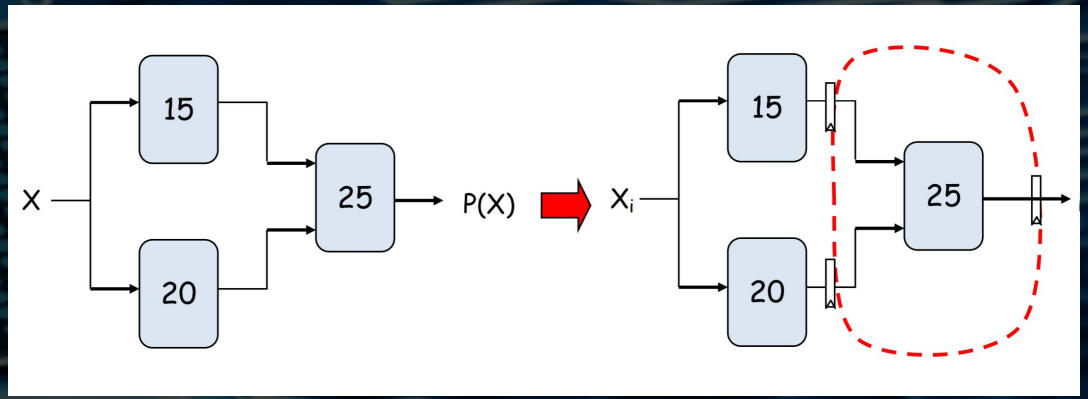


Controls the current program counter which points to the currently executing instruction



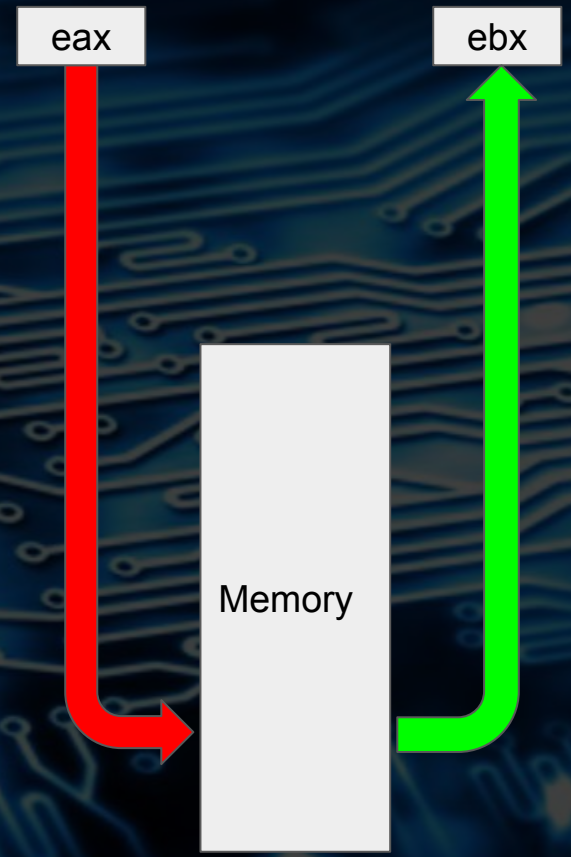
Extensions

- **Pipelining**



Extensions

- **Pipelining**
- **Additional Instructions**
  - **mov (%eax), %ebx**

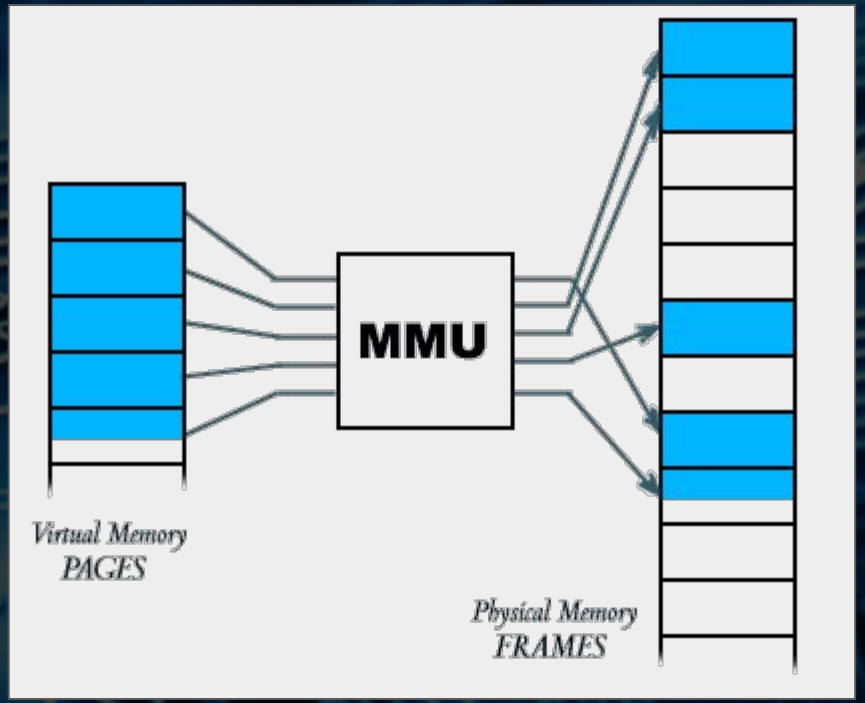


Extensions

- **Pipelining**
- **Additional Instructions**
  - **mov (%eax), %ebx**
- **Complex Instruction Set**
  - **pusha**

Extensions

- Pipelining
- Additional Instructions
  - mov (%eax), %ebx
- Complex Instruction Set
  - pusha
- Additional Hardware Components
  - MMU



Extensions

- **Pipelining**
- **Additional Instructions**
  - **mov (%eax), %ebx**
- **Complex Instruction Set**
  - **pusha**
- **Additional Hardware Components**
  - **MMU**
- **Additional Optimizations**

Additional Goals from Presentation

- Our demonstration strategy:
  - Merge sort demo
- Move pipelining to stretch goal
- Add console interface
  - Use switches to input command or data set

Timeline

	Week 1 (11/7 - 11/14)	Week 2 (11/14 - 11/21)	Week 3 (11/21 - 11/28)	Week 4 (11/28 - 12/5)	Week 5 (12/5 - 12/10)
Control Logic	B	B	Integration (B+Q)		
ALU	Q	Q			
Register File		Q			
Memory (Instruction and data)		B			
PC		Q			
Pipelining				B	
Extensions					Q