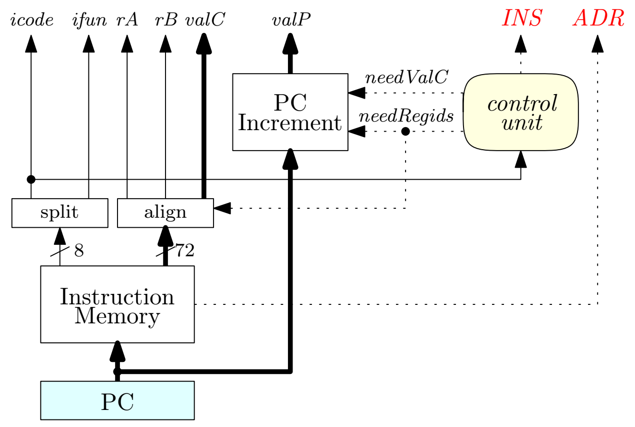
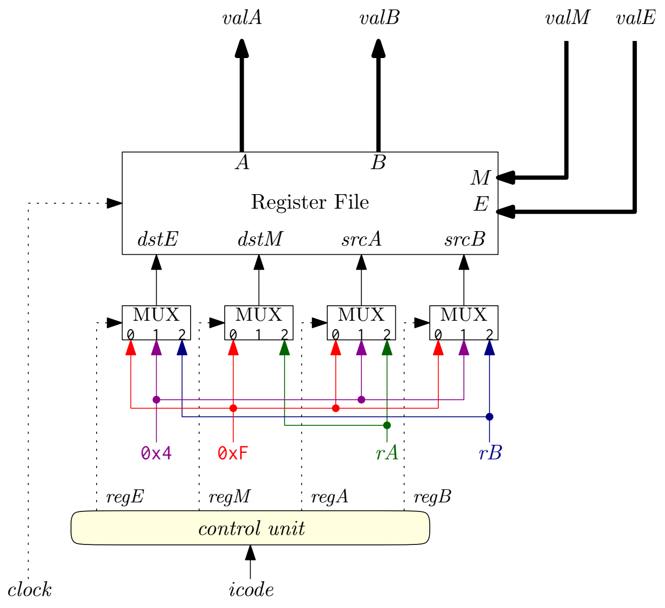
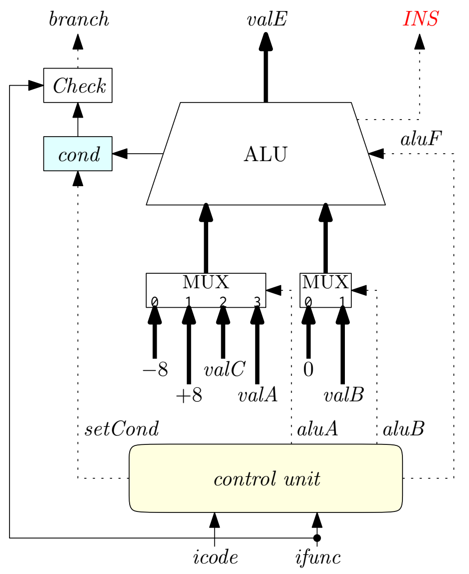
Computation Steps in Single-Cycle Implementation of Y86

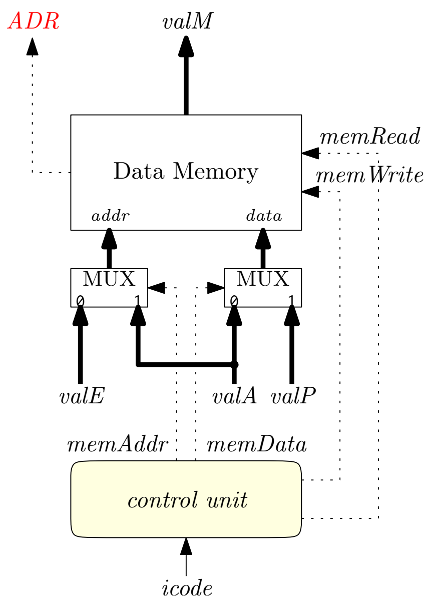
Stage	Step	`hlt`	`nop`	`rrmov`	`irmov`	`rmmov`	`mrmov`
Fetch	*icode, ifun*	$icode{:}ifun \leftarrow M_1[PC]$	$icode{:}ifun \leftarrow M_1[PC]$	$icode{:}ifun \leftarrow M_1[PC]$	$icode{:}ifun \leftarrow M_1[PC]$	$icode{:}ifun \leftarrow M_1[PC]$	$icode{:}ifun \leftarrow M_1[PC]$
	rA, rB			$rA{:}rB \leftarrow M_1[PC+1]$	$rA{:}rB \leftarrow M_1[PC+1]$	$rA{:}rB \leftarrow M_1[PC+1]$	$rA{:}rB \leftarrow M_1[PC+1]$
	valC				$valC \leftarrow M_8[PC+2]$	$valC \leftarrow M_8[PC+2]$	$valC \leftarrow M_8[PC+2]$
	valP	$valP \leftarrow PC+0$	$valP \leftarrow PC+1$	$valP \leftarrow PC+2$	$valP \leftarrow PC+10$	$valP \leftarrow PC+10$	$valP \leftarrow PC+10$
Decode	*valA*			$valA \leftarrow R[rA]$		$valA \leftarrow R[rA]$	
	valB					$valB \leftarrow R[rB]$	$valB \leftarrow R[rB]$
Execute	*valE*			$valE \leftarrow 0 + valA$	$valE \leftarrow 0 + valC$	$valE \leftarrow valB + valC$	$valE \leftarrow valB + valC$
	cond						
Memory	*valM*					$M_8[valE] \leftarrow valA$	$valM \leftarrow M_8[valE]$
Write back	*dstE*			$R[rB] \leftarrow valE$	$R[rB] \leftarrow valE$		
	dstM						$R[rA] \leftarrow valM$
PC Update	*PC*	$PC \leftarrow valP$	$PC \leftarrow valP$	$PC \leftarrow valP$	$PC \leftarrow valP$	$PC \leftarrow valP$	$PC \leftarrow valP$

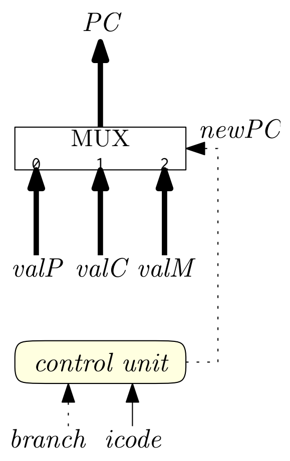
Stage	Step	`OP`	`jXX`	`call`	`ret`	`push`	`pop`
Fetch	*icode, ifun*	$icode{:}ifun \leftarrow M_1[PC]$	$icode{:}ifun \leftarrow M_1[PC]$	$icode{:}ifun \leftarrow M_1[PC]$	$icode{:}ifun \leftarrow M_1[PC]$	$icode{:}ifun \leftarrow M_1[PC]$	$icode{:}ifun \leftarrow M_1[PC]$
	rA, rB	$rA{:}rB \leftarrow M_1[PC+1]$				$rA{:}rB \leftarrow M_1[PC+1]$	$rA{:}rB \leftarrow M_1[PC+1]$
	valC		$valC \leftarrow M_8[PC+1]$	$valC \leftarrow M_8[PC+1]$			
	valP	$valP \leftarrow PC+2$	$valP \leftarrow PC+9$	$valP \leftarrow PC+9$	$valP \leftarrow PC+1$	$valP \leftarrow PC+2$	$valP \leftarrow PC+2$
Decode	*valA*	$valA \leftarrow R[rA]$			$valA \leftarrow R[4]$	$valA \leftarrow R[rA]$	$valA \leftarrow R[4]$
	valB	$valB \leftarrow R[rB]$		$valB \leftarrow R[4]$	$valB \leftarrow R[4]$	$valB \leftarrow R[4]$	$valB \leftarrow R[4]$
Execute	*valE*	$valE \leftarrow valB\ op\ valA$		$valE \leftarrow valB + (-8)$	$valE \leftarrow valB + (+8)$	$valE \leftarrow valB + (-8)$	$valE \leftarrow valB + (+8)$
	cond	$cond \leftarrow Cond(valE)$	$b \leftarrow C(cond, ifun)$				
Memory	*valM*			$M_8[valE] \leftarrow valP$	$valM \leftarrow M_8[valA]$	$M_8[valE] \leftarrow valA$	$valM \leftarrow M_8[valA]$
Write back	*dstE*	$R[rB] \leftarrow valE$		$R[4] \leftarrow valE$	$R[4] \leftarrow valE$	$R[4] \leftarrow valE$	$R[4] \leftarrow valE$
	dstM						$R[rA] \leftarrow valM$
PC Update	*PC*	$PC \leftarrow valP$	$PC \leftarrow valC$ if b else $valP$	$PC \leftarrow valC$	$PC \leftarrow valM$	$PC \leftarrow valP$	$PC \leftarrow valP$


icode
ifun
rA
rB
valC
valP
INS
ADR
PC Increment
needValC
needRegids
control unit
split
align
8
72
Instruction Memory
PC


valA
valB
valM
valE
A
B
M
E
Register File
dstE
dstM
srcA
srcB
MUX
0 1 2
0x4
0xF
rA
rB
regE
regM
regA
regB
control unit
clock
icode


branch
valE
INS
Check
cond
ALU
aluF
MUX
0 1 2 3
0 1
−8
+8
valC
valA
0
valB
setCond
aluA
aluB
control unit
icode
ifunc


ADR
valM
Data Memory
memRead
memWrite
addr
data
MUX
0 1
valE
valA
valP
memAddr
memData
control unit
icode


PC
MUX
0 1 2
newPC
valP
valC
valM
control unit
branch
icode