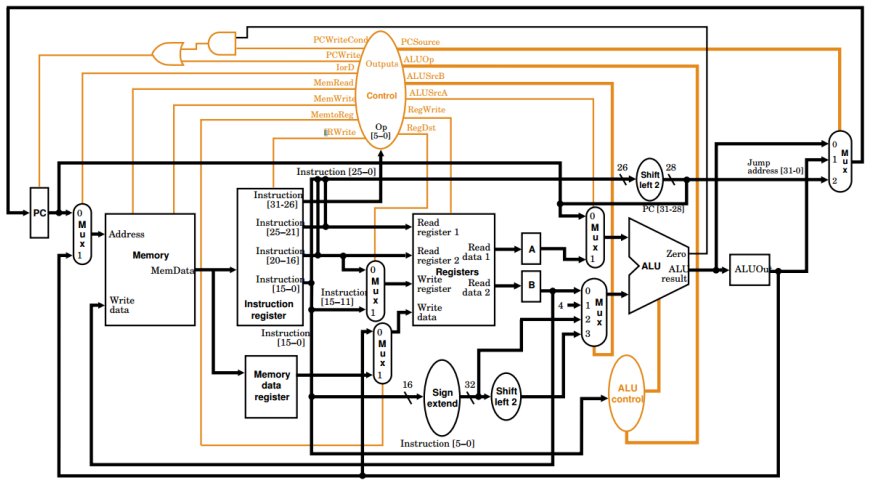
1. [6] Si consideri la CPU in figura mentre esegue il codice a lato: si specifichi il contenuto (quando determinabile), ad ogni ciclo di clock, dei registri: A, B, ALUOut e MDR (in formato esadecimale). (OpCodes: addi:8, lw:36, sw:44)

```
0x8F8: addi $8, $0, 8
       sw $8, 80($8)
       lw $10, 88($0)
```



2. [4] Disegnare la struttura circuitale interna del moduli: "Sign extend" e di entrambi i circuiti "Shift left 2" presenti nella CPU in figura.

3. [8] In una CPU con frequenza di clock 1 GHz (=10^9 Hz) e bus dati di 32 bit, una periferica trasferisce informazioni al ritmo di 6 milioni di byte/s mediante DMA, trasferendo blocchi di 4000 byte per volta in modo sincrono sul bus (1 parola/ciclo di clock). Supponendo che siano necessari 1500 cicli di clock per l'inizializzazione e altri 500 per la terminazione di ogni blocco DMA, calcolare:

a) quanti cicli di clock sono necessari per ogni blocco;

b) la percentuale di tempo di CPU dedicato alle operazioni di DMA durante il trasferimento.

c) la percentuale di tempo di CPU dedicata al trasferimento dati dalla stessa periferica, se si utilizzassero chiamate di interrupt anziché il DMA, supponendo che ogni chiamata di interrupt richieda 40 cicli di clock e trasferisca una parola.

4. [3] Elencare e descrivere le soluzioni architetturali mediante le quali si rende possibile l'accesso alle periferiche di I/O a un programma Assembly.

5. [5] Si vogliono incrementare le prestazioni di un calcolatore sostituendo l'unità a dischi con una a stato solido, grazie alla quale il "transfer rate" passa da 25 a 525 MB/s (si considerino i tempi di accesso trascurabili). In seguito a questa sostituzione, un programma passa da un tempo di elaborazione di 3 ore e mezza a 42 minuti. Qual è la percentuale di tempo dedicata al trasferimento su disco da tale programma? Quant'è il tempo di elaborazione minimo raggiungibile per tale programma?

6. [5] Mostrare, mediante gli opportuni frammenti di codice MIPS, come allocare a) staticamente e b) dinamicamente un array di 1M (2^{20}) interi, inizializzando ciascun elemento degli array al valore del suo indice (cioè: `A[i] = i`).

System calls

	codice ($v0)	argomenti	risultato
print_int	1	$a0	
print_float	2	$f12	
print_double	3	$f12	
print_string	4	$a0	
read_int	5		$v0
read_float	6		$f0
read_double	7		$f0
read_string	8	$a0,$a1	
sbrk	9	$a0	$v0
exit	10		

Registri MIPS

0	zero		24-25	t8-t9
1	at		26-27	k0-k1
2-3	v0-v1		28	Gp
4-7	a0-a3		29	Sp
8-15	t0-t7		30	s8
16-23	s0-s7		31	Ra