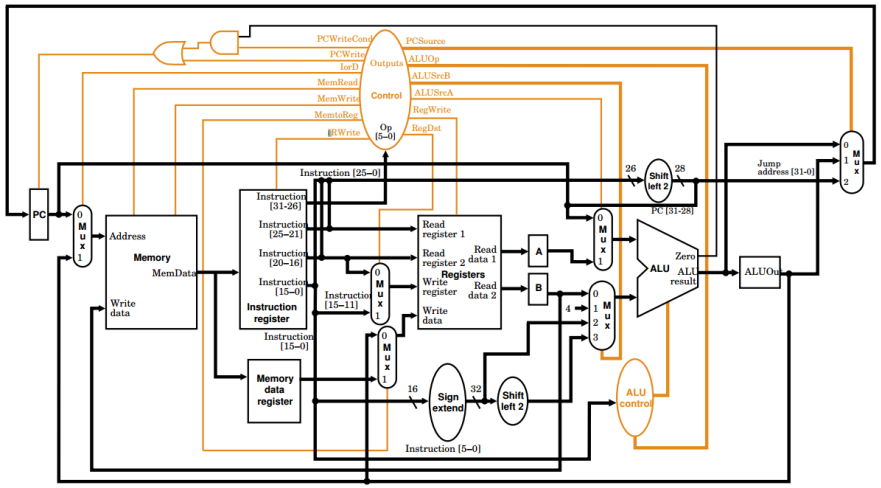
1. [5] Considerando la CPU in figura, determinare il contenuto dei registri **PC, IR, MDR** e **ALUout**, per ogni ciclo di clock necessario ad eseguire il frammento di codice a lato.
(Opcodes: **`addi`**=8, **`sw`**=44, **`bne`**=5)

```
0xCCC: addi $6,$0, -12
       sw   $6, 32($0)
       bne $6, $0, -12
```



2. [5] In una CPU con frequenza di clock 1 GHz e bus dati di 64 bit, una periferica trasferisce informazioni mediante DMA, trasferendo blocchi di 16000 byte per volta in modo sincrono sul bus (1 parola/ciclo di clock). Supponendo che siano necessari 1200 cicli di clock per l'inizializzazione e altri 800 per la terminazione di ogni blocco DMA, calcolare:

a) la velocità di trasferimento periferica/memoria ottenibile con un'occupazione percentuale di tempo di CPU pari a 1%.

b) la percentuale di tempo di CPU dedicata al trasferimento dati dalla stessa periferica alla stessa velocità, se si utilizzassero chiamate di interrupt anziché il DMA, supponendo che ogni chiamata di interrupt richieda 200 cicli di clock e trasferisca 4 parole alla volta.

3. [5] Definire il fenomeno della criticità di salto in CPU pipeline e descrivere in dettaglio la tecnica di *branch prediction* per gestirla.

4. [5] Un processore caratterizzato da uno spazio di memoria indirizzabile di 16 GByte e un bus dati di 64 bit viene dotato di una memoria a mappatura diretta di capacità totale di 2 MByte e linee di 32 parole.
a) Disegnare lo schema circuitale dettagliato di tale memoria;
b) calcolare il numero di bit necessari per ciascuna linea (includendo tutti i campi);
c) determinare l'indirizzo di memoria principale (in esadecimale) corrispondente a: byte offset=3, word offset=3, index=3 e tag=3.

5. [6] Rappresentare il contenuto, byte per byte in formato esadecimale, e gli indirizzi corrispondenti, della zona di memoria che viene modificata a seguito dell'esecuzione del frammento di codice a lato, considerando un'organizzazione della memoria di tipo "little endian" (si ricorda che il codice ASCII di "A"=65).

```
.data 0x4A0
.byte –9
.word –15, 0x15
.space 6
.align 2
.asciiz "ABC"
```

6. [5] Mostrare, mediante gli opportuni frammenti di codice MIPS, come allocare a) staticamente e b) dinamicamente un array di 2048 interi, inizializzando ciascun elemento degli array al valore doppio del suo indice (cioè: **`A[i] = 2i`**).

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`