Julio Soto  
CSCI 5593  
05FEB19

1.

I followed the steps on the instructions and created the files necessary on the CSE network, but am unable to view the webpage when attempting to follow the link provided.

2.*A*

|  |  |
| --- | --- |
| **Register** | |
| Load | R1, A |
| Load | R2, B |
| Add | R3, R1, R2 |
| Store | Store R3, C |

2. *B*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Description** | Operation Code (op) | Source Register (rs) | Source Register 2 (rt) | Destination Register (rd) | Shift Amount (shift) | Function Code (funct) |
| **Size** | 6 bits | 5 bits | 5 bits | 5 bits | 5 bits | 6 bits |
| **Reason for size** | 6 bits are needed to store the 50 already stored commands/instructions | 5 bits are needed to represent the number, thus requiring 5 bits | | | Amount by which the source register is shifted | The parameter contains the necessary control codes to differentiate the different instructions; which are 6 bits long. |

2. *C*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description** | Operation Code (op) | Source Register (rs) | Source Register 2 (rt) | Address |
| **Size** | 6 bits | 5 bits | 5 bits | 16 bits |
| **Reason for size** | 6 bits are needed to store the 50 already stored commands/instructions | 5 bits are needed to represent the number, thus requiring 5 bits | | In order to facilitate base + displacement; provides address where the data has to be stored to or to be loaded from. |

3. *A*

|  |  |
| --- | --- |
| **Accumulator** | |
| Load | A |
| Add | B |
| Store | C |

3. *B*

|  |  |  |  |
| --- | --- | --- | --- |
| |🡨 32 bits 🡪| | | | |
| **Description** | Index address mode - X | Operation Code (op) | Address |
| **Size** | 1 bit | 7 bits | 24 bits |
| **Reason for size** | Address field + X | Capable of performing 128 operations; each operation must be 7 bits | 2^24 words of memory are used; each word is 24 bits; depending on the opcode |

4.

|  |  |
| --- | --- |
| **Instruction Class** | **Spice** |
| Arithmetic | 50% |
| Data Transfer | 41% |
| Conditional Branch | 8% |
| Jump | 1% |

*A*  
Data / (Data + Instructions) = 0.41 / (0.41+IC) = 0.41 / 1.41 = 29%

*B*  
Reads / (Data + Instructions) = (IC + (2/3 \* 0.41)) /(.41+IC) = 1.27 / 1.41 = 90%

5.

Literature Review of  
Power-aware Job Scheduling on Heterogeneous Multicore Architectures  
By Matteo Chiesi, Luca Vanzolini, Claudio Muci, Eleonora Franchi Scarselli, and Roberto Guerrieri

The research in this paper provides a power-aware scheduling algorithm based on efficient distribution of the computing workload to the resources on heterogeneous CPU-GPU architectures. As a background, supercomputers that have been designed consume a lot of power in order to conduct their operations during periods of high computation or “worst-case scenario,” as the paper refers to these moments in time. While supercomputers were designed for high computing capacity, they did not foresee the cooling and power supply issues to surpass the initial capital investment. Thus, requiring them to come up with a solution to reducing power consumption while still having the computer run at peak performance. Power consumption during concurrent execution increases but performance is increased. However, power and performance decreases when concurrency is decreased, which is also not a viable solution since high performance was intended for such a machine.

A job-level scheduling algorithm was created by the authors of this paper to limit the worst-case power condition that would enable to the computer to reduce power consumption below a threshold specified in order maintain efficient performance during concurrent execution for a heterogenous system. Three contributions were made during this research, and they are as follows: 1. A low-cost measurement system was developed to profile the power of the jobs running on heterogeneous computer architectures, 2. A power-aware scheduling algorithm was developed to manage the resources of the several computer nodes, and 3. A quantitative analysis was conducted to demonstrate the algorithm’s effectiveness on reducing power cost during concurrent job execution.

Creating this new power-aware scheduling algorithm produced a peak power reduction of as much as 10 percent during concurrent execution the experiments, compared to a system without a power-aware policy. This was done by every so slightly increasing the execution time, thus making it possible to reduce the power supply cost. However, the team mentions the limitations these results depend on the target architecture of this work, and that execution of this algorithm on different architectures could lead to different peak power values. The process followed by the research would need to be conducted on any new system in order to characterize the power consumption and maximize performance during peak execution of resources.

**Reference:**

[1] M. Chiesi, L. Vanzolini, C. Mucci, E. F. Scarselli, and R. Guerrieri, “Power-Aware Job Scheduling on Heterogeneous Multicore Architectures,” *ieeexplore-ieee-org.aurarialibrary.idm.oclc.org*, Mar-2015. [Online]. Available: https://ieeexplore-ieee-org.aurarialibrary.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=6782408. [Accessed: 03-Feb-2019].